

N O T I C E

THIS DOCUMENT HAS BEEN REPRODUCED FROM
MICROFICHE. ALTHOUGH IT IS RECOGNIZED THAT
CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED
IN THE INTEREST OF MAKING AVAILABLE AS MUCH
INFORMATION AS POSSIBLE

Two Dimensional Recursive Digital Filters
for Near Real Time Image Processing

Final Report

NASA Grant N.G 5406

Period September 15, 1979 to December 30, 1980

By

Dr. Winser E. Alexander - Principal Investigator

Dr. David Olson

Mr. Earnest Sherrod

(NASA-CR-164028) TWO DIMENSIONAL RECURSIVE
DIGITAL FILTERS FOR NEAR REAL TIME IMAGE
PROCESSING Final Report, 15 Sep. 1979 - 30
Dec. 1980 (North Carolina Agricultural and
Technical) 194 p HC A09/MF A01

N81-19398

Unclassified
41619



Department of Electrical Engineering

North Carolina Agricultural and Technical
State University

Greensboro, North Carolina 27411

Table of Contents

	Page
Abstract	2
1.0 Introduction	2
2.0 Background	3
3.0 Mathematical Theory	5
4.0 Stability Analysis	8
5.0 Synthesis	9
6.0 Implementation	11
6.1 Implementation Considerations	11
6.2 Transient Response	13
6.3 Implementation Algorithms	14
7.0 Applications	15
7.1 Dynamic Range Compression	15
7.2 Subjective Image Processing	16
7.3 Bandwidth Optimization	17
7.4 Interpolation	17
7.5 Image Registration, Classification and Evaluation	18
8.0 Image Processing Facilities	19
9.0 Image Processing Results	20
12.0 References	24
Appendices	
A. Investigation of Alternative Realization Techniques	
B. Implementation Consideration for Two Dimensional Recursive Digital Filters with Product Separable Deminonators	
C. Imagery Programs	
D. Imagery Program Files	
E. Papers	

Abstract

This program was specifically oriented toward the demonstration of the feasibility of using two dimensional recursive digital filters for subjective image processing applications that require rapid turn around. The concept of the use of a dedicated minicomputer for the processor for this application was also to be demonstrated. The minicomputer used was the HP1000 series E with a RTE II disc operating system and 32K words of memory. A Grinnel 256 X 512 X 8 bit display system was used to display the images.

Sample images were provided by NASA Goddard on a 800 BPI, 9 track tape. Four 512 X 512 images representing 4 spectral regions of the same scene were provided. These images were filtered with enhancement filters developed during this effort and returned to NASA Goddard for further analysis.

1.0 INTRODUCTION

The goal of this program was to develop algorithms to be used in the laboratory on a near real time basis to enhance the capability of a trained observer to obtain geologically interesting information from Landsat satellite imagery. Each Landsat image is recorded with 4 separate spectral bands: 3 in the visible and 1 in the infrared. Thus each scene to be processed is composed of 4 images. Four such images of a scene of interest was provided by NASA Goddard as test images for the program. Each image was provided with 512 rows of 512 pixels per row and 8 bits per pixel.

The objectives of the program were to develop software to implement previously designed two dimensional recursive digital filters on the Department of Electrical Engineering's HP1000 computer system [3]. These filtering algorithms were to be used in an evaluation of the feasibility of their use to

aid the extraction of geologically interesting data from Landsat images. The sample images were to be processed and provided to NASA Goddard for analysis and evaluation.

It was not an objective of this program to approach near real time performance because there was no opportunity to optimize the system hardware for this purpose. A pipeline or array processor would have to be added to improve the computational capability of the system. However, the performance of the system could be used to assess feasibility of further research and development in this area.

2.0 BACKGROUND

Digital filters can be classified as being of two basic types: transform domain filters and time or spatial domain filters. The filtering process is performed in the frequency or transform domain with transform domain filters. The transforms of the signal to be filtered and the impulse response of the desired filter are multiplied to form the transform of the output signal. The inverse transform of the result provides the filtered output signal. Thus any filtering operation requires two transform operations and a multiplication operation. The Discrete Fourier (DFT) is commonly used for most transform domain filtering operations. The Fast Fourier Transform (FFT) algorithm provides a means of implementing the DFT in a computationally efficient manner. Time or spatial domain digital filters do not require a transform process. The filtering is done by taking a weighted average of input and past output values to compute the current output.

There are basically two types of image enhancement: subjective image enhancement and image correction. In subjective image enhancement, the object is to process the image in such a way as to make an improvement in its

appearance or ability to transfer information in some way. If this type of image enhancement is of interest, the user should have available a multitude of general purpose image processing functions. These would include (but not be limited to) low pass filters, high pass filters, low and high frequency enhancement filters, line enhancement filters and line suppression filters. Most of these filtering operations can effectively be accomplished by two dimensional spatial domain digital filters. There is no inherent need to obtain the DFT in the filtering process.

Spatial domain filtering using digital recursive filters offers savings in computation time and core requirements over the use of transform methods to achieve the same filtering process [1]. This is accomplished for many filtering operations with no sacrifice in the quality of the output. Therefore, it is advantageous to use recursive digital filters for those functions for which appropriate filtering algorithms can be developed.

Spatial domain filtering using digital nonrecursive filters offer advantages over both recursive digital filters and FFT digital filters when the number of filter coefficients are relatively small. However, the filters available that meet this requirement are limited. For this reason, nonrecursive digital filters can only be applied to special cases for use in near real time processing. In general, it requires a greater number of coefficients to realize a particular impulse response for nonrecursive digital filters than for recursive digital filters.

Image correction requires a much more complicated filtering process in general than does subjective image processing. The object is to make corrections for distortion, blurring, smearing, etc., that occurred while the image was being formed. This requires the approximation of a filtering function

which is the inverse of the modulation transfer function (MTF) of the imaging process. It is usually necessary to make modifications for the phase as well as the magnitude of the MTF. The resulting filtering requirements are often very complicated and the design of the required digital filter is not a trivial process.

The application of the two dimensional recursive digital filters to image processing and other two dimensional data has been hampered by two problems: stability and synthesis. The synthesis problem is the problem of expressing the two dimensional Z-Transform of the desired impulse response in closed form and thus determining the filter coefficients. The stability problem is important because the recursive filter requires feedback of past output values and therefore can become unstable. Research results obtained on both of these problems by the authors have demonstrated that two dimensional recursive digital filters are very practical for image processing applications [2,3].

3.0 MATHEMATICAL THEORY

The theoretical basis for the two dimensional ZW-Transform [4] involves the theory for sample data systems. Given discrete samples of a two dimensional function, $f(x,y)$ with sampling increments X and Y respectively, the ZW-Transform for the function is defined by

$$F(z,w) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(mX, nY) z^{-m} w^{-n} \quad (3.1)$$

If the function is an image, then the problem can be set up so that m and n have no negative values and the range of m and n is finite. We further restrict the problem to the case where X and Y are constants. Then, if we use the notation $f(m,n)$ to represent $f(mX, nY)$, we have

$$F(z,w) = \sum_{m=0}^M \sum_{n=0}^N f(m,n) z^{-m} w^{-n} \quad (3.2)$$

as the ZW-Transform for the image function, $f(m,n)$, which has $(M + 1)$ columns and $(N + 1)$ rows.

Consider the case where we have an input image with samples $f(m,n)$ and we wish to filter this image to obtain an output image with corresponding samples, $g(m,n)$. The samples of the impulse response of the desired filter are given by $h(m,n)$. The range of m and n for the output is the same as for the input. Thus, the ZW-Transform of $g(m,n)$ is given by

$$G(z,w) = \sum_{m=0}^M \sum_{n=0}^N g(m,n) z^{-m} w^{-n} \quad (3.3)$$

If we restrict the impulse response such that m and n cannot be negative (a causal system), we can write the ZW-Transform for the impulse response as

$$H(z,w) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} h(m,n) z^{-m} w^{-n} \quad (3.4)$$

In general, the ZW-Transform for the impulse response is an infinite series. In order to implement the spatial domain filter, we must find a closed form expression for $H(z,w)$ such that

$$H(z,w) = \frac{\sum_{J=0}^L \sum_{K=0}^L a_{JK} z^{-J} w^{-K}}{\sum_{J=0}^L \sum_{K=0}^L b_{JK} z^{-J} w^{-K}} \quad (3.5)$$

Some of the coefficients, a_{JK} and b_{JK} may be zero. The convolution property of the ZW-Transform gives the relationship resulting from the convolution of $f(m,n)$ and $h(m,n)$ which is the filtering process

$$G(z,w) = H(z,w)F(z,w) \quad (3.6)$$

If we use the closed form of $H(z,w)$ and restrict b_{00} to be equal to one and write the resulting equation for a single output value $g(m,n)$, we obtain the difference equation for the causal filter

$$g(m,n) = \sum_{J=0}^L \sum_{K=0}^L a_{JK} f(m-J, n-K) - \sum_{\substack{J=0 \\ J+K>0}}^L \sum_{K=0}^L b_{JK} g(m-J, n-K) \quad (3.7)$$

If L is relatively small (in practice, L is usually less than 10 for recursive digital filters), equation (3.7) represents a very efficient algorithm for filtering images. Equations (3.5) and (3.7) may also represent a nonrecursive filter if all b_{JK} except b_{00} are equal to zero.

4.0 STABILITY ANALYSIS

Nonrecursive digital filters are inherently stable. Since there is no feedback of past output values, the impulse response has finite duration. Each output value is a finite sum which is always bounded if the input is bounded.

The stability problem for one dimensional digital recursive filters is straight forward. The roots of the denominator polynomial in the closed form of the one dimensional Z-Transform for the filter impulse response function must have magnitudes less than one. Stability analysis is therefore reduced to finding roots of nth degree polynomials with real, constant coefficients [5]. Stability analysis is not straight forward for the two dimensional problem because a two variable polynomial is not generally factorable into distinct roots. When the polynomial in the denominator of the two dimensional Z-Transform of the impulse response is factorable into distinct roots, the stability analysis procedure is the same as for the one dimensional problem.

The two dimensional stability problem is very complicated if the polynomial in the denominator is not factorable into distinct roots [6]. Efforts by other researchers have been directed toward examining regions of roots for two variable polynomials. The developed procedures are computationally feasible only for very simple filters. An alternate method of assessing stability for one dimensional digital recursive filters is to make a state space representation of the filter [7]. Then the filter is stable if the eigenvalues of the state transition matrix all have magnitudes less than one. Previous research has been directed toward developing the two dimensional equivalent of this procedure [2]. A pseudo-state variable representation is chosen because of difficulties in finding a true state space representation [8]. This difficulty is caused by the bivariance of the transfer function and by its causality. The

resulting matrix equation has two pseudo-state transition matrices.

Previous results have shown that the corresponding filter is unstable if any of the eigenvalues of either of these matrices have magnitudes greater than or equal to one or if any of the eigenvalues of the matrix sum have magnitudes greater than or equal to one. Reprints of papers presenting these results are included as in [2].

In practice, these constraints have been found to be very useful in that all tested filters that were known to be unstable were identified as such by the procedure. Conversely, all filters which were known to be stable met the criteria for stable filters and were not identified as unstable.

5.0 SYNTHESIS

The synthesis of nonrecursive digital filters is not a major effort in the proposed research. Several simple nonrecursive digital filter designs may be found in the literature [9]. It would be appropriate to evaluate these designs with regard to application to near real time processing of Landsat satellite data. However, this was not a part of this program.

Often it is possible to express a desired two dimensional recursive digital filter as the product or sum of two one dimensional digital filters. That is the two dimensional Z-Transform of the digital recursive filter can be expressed as the product or sum of two one dimensional Z-Transforms. In either case, the two dimensional synthesis problem is reduced to the synthesis of two one dimensional filters. However, it is not possible to design sum separable or product separable digital recursive filters for all applications. For these applications, the design of the required two dimensional digital recursive filter is considerably more complicated.

Many imaging systems have a natural circular symmetry. In general, the optical transfer function of a circularly symmetric imaging system is circularly symmetric. Also, it is usually desirable to perform image processing where the processing is uniform with respect to direction. The natural consequence is that filters with circularly symmetric impulse response functions are generally very desirable for image processing. The relationship between circular symmetry of the impulse response and the frequency response dictates that the design requirement is for these filters to have a circularly symmetric frequency response [10].

Previous research efforts have led to a synthesis technique which yields two dimensional recursive lowpass, highpass, low frequency boost and high frequency boost recursive digital filters that are very close to being circularly symmetric when the cutoff frequencies are approximately one half the Nyquist frequency [3,11]. Some degradation is observed as the cutoff frequency approaches either the Nyquist frequency or zero.

In the design procedure, the squared magnitude characteristic of the desired circularly symmetric filter is chosen in the Laplace Transform domain. The bilinear transformation is then used to map the squared magnitude characteristic into the two dimensional Z-Transform domain. The pseudo-state space representation for the corresponding two dimensional Z-Transform is formed. The eigenvalues of the matrix sum of the two pseudo-state transition matrices are obtained. These eigenvalues occur in reciprocal pairs. The eigenvalues with magnitudes less than one are then used as roots of a denominator polynomial with distinct roots to form the two dimensional Z-Transform of the desired filter.

Note that this design procedure always ensures a stable filter. Stability analysis is simple because the denominator of the ZW - Transform is a product separable. Also note that no restrictions are placed on the numerator polynomial. That is, it is not necessary for the numerator polynomial to either be product separable, sum separable or minimum phase. Examples of stable two dimensional recursive filter designs are given in [12].

Another problem of interest in image processing is to filter with a one dimensional filter with the orientation of the filter specified and independent of the sampling direction. This type of filter would be useful for enhancing or suppressing linear features, for system noise suppression or for image correction (i.e., linear smear). However, any one dimensional digital recursive filter which is rotated becomes a two dimensional digital recursive filter with associated problems in stability and synthesis. Constraints with regard to stability of rotated digital filters have been developed [13,14]. However, the problems associated with the actual synthesis of rotated recursive digital filters have not been adequately addressed. This is a problem of interest to this research program. However, it was not pursued during this effort.

6.0 IMPLEMENTATION

6.1 Implementation Considerations

Recursive digital filters have many very desirable features that make them advantageous for real time or near real time image processing applications. In the practical application of recursive digital filters, only a small number of rows of the image to be processed are required to be stored in the computer at one time. Three rows of storage plus three rows of storage for each pair of complex poles in the transfer function to be realized are required. Thus a filter with two poles and two zeros would require the storage of the equivalent

of six rows of the input image. A filter with four poles and four zeros would require the storage of the equivalent of nine rows of the input image.

Most image filtering requirements may be met with a filter having no more than four zeros and four poles. Therefore, an algorithm which allows up to four zeros and four poles is practical. Such a filter would still require only slightly more than 9216 storage locations to filter a 1024 by 1024 image. Some additional storage would be required to store the code for the algorithm including its interface to data handling algorithms. Thus it is quite feasible to use recursive digital filters to filter images up to 1024 by 1024 using a 16 bit minicomputer with only 64k words of storage. If in addition a pipeline or array processor is used to implement the recursive digital filter itself, extremely fast processing can be accomplished. In fact, the processing time may be limited by the time required to transfer the data from and back to the storage medium during the actual filtering process.

Recursive digital filters typically require fewer data transfer operations to filter a given image than FFT filters. This is particularly true for very large images. The FFT filtering algorithm requires that the image be transformed by row and then by column. If the image is too large to fit in the computer at one time, the FFT algorithm becomes inconvenient to use for filtering images. One method commonly used to overcome this difficulty is to filter the image in blocks which are small enough to fit into the computer and then fit these filtered blocks back together to form the output image. Considerable overlap of these blocks is required to avoid artifacts due to periodic convolution. Average levels between blocks also have to be adjusted to avoid a checkerboard effect. Another method commonly used is to transform the image by rows, transpose the image and then transform the image by columns [15].

This procedure adds two transpose operations to each filtering operation. The result is that in the practical use of filtering large images, recursive digital filters are very significantly more efficient and require far less time to implement than FFT filters.

Recursive digital filters inherently have nonlinear phase characteristics. This is true because of feedback of past output values. However, linear phase can be obtained by filtering the image twice [3]. The image is filtered starting from the first row, first pixel and ending with the last row, last pixel. Then the image is filtered backward starting with the last row, last pixel and ending with the first row, first pixel. The result is a filter transfer characteristic which is the magnitude squared of the original characteristic. Thus, the filter with four poles and four zeros effectively has eight poles and eight zeros and linear phase when this procedure is used.

6.2 Transient Response

The use of past values of the output to compute the current output value results in the equivalent of long term storage of information about past inputs for recursive digital filters. Thus, such filters have an infinite impulse response (IIR). In addition, the beginning of each scan line in an image represents a transient which can cause very undesirable results if the implemented filter has a long term transient response. If this situation is not handled properly, then two dimensional recursive digital filters will give very poor results. This is particularly true for high frequency boost or highpass filters.

The approach used to minimize this problem is to place the filter in a stable state with an assumed input within the range of the image data. The best assumed input would be the expected value of the input image intensity.

However, this is usually not available. An approximation is obtained by averaging the intensity values of the middle row of the image. The final value theorem [5] is then used to determine the stable state for each of the output stages for the filter. The expected values approximation is then used as the initial condition input for each scan line and the stable state output is used as the initial condition output for each filter stage. Thus, if the initial input is the same as the assumed initial condition, then no transient response occurs.

In practice, the procedure outlined above is simple to implement and add very computations to the filtering process. However, additional improvement can be obtained by extending the image by using a reflection of future pixel values. Typically as few as 5 values produces very good results such that no transient response artifacts may be observed with most filter designs.

6.3 Implementation Algorithms

Equation 3.7 provides the fundamental algorithm for the two dimensional recursive digital filter. A straight forward approach is to implement the filter directly as provided. However, consideration must be given to roundoff error (the HP1000 computer uses 32 bit floating point arithmetic) and computational efficiency. In addition, the use of complex numbers should be avoided. Therefore, the fundamental stage for the filters was selected to be a second order stage with L equal to 2 in (3.7). Higher order filters may be implemented using multiple stages. This also allows combinations of filters such as a low pass filter for noise removal and a high frequency boost filter for edge enhancement.

In writing the actual algorithm, care was taken to use one dimensional arrays and to avoid transferring data between arrays when possible. Thus a computationally efficient algorithm was developed.

The fact that the HP1000 series E uses a software floating point arithmetic processor and only has a total of 32 K words (64 K bytes) of memory provided a severe hardware limitation. This system has just recently been upgraded to the series E RTE-IVB with an additional 64 K words of memory and a hardware floating point processor. Thus the performance of the image processing software should be very significantly improved with these hardware changes.

In addition to the implementation considerations described above, research was conducted with regard to devising special algorithms which can be used in parallel or pipeline architectures to approach real time image processing. Appendix A and B provide details on this effort. Appendix C and D gives documentation of the software developed.

7.0 APPLICATIONS

7.1 Dynamic Range Compression

Electro-optical sensors respond to reflected or emitted radiation. A typical electro-optical imaging system uses a single detector or an array of detectors in a scanning mode to form the image. If the signal of interest is the reflected radiation such as is the case for visible imaging systems, the detected signal is made up of two components: the illumination component and the reflection component. Infrared sensors typically detect radiation emitted by objects. It is typical that the available dynamic range of electro-optical imaging systems is several orders of magnitude. On the other hand, display systems are usually limited to at most two orders of magnitude and human observers can only detect approximately 50 different intensity levels [16].

Therefore, it is not possible to directly display all information obtained in many images.

The illumination component of optical images or the overall background radiation for infrared images generally has low spatial frequency content but may have a wide dynamic range [17]. This is the case where shadows exist in optical imagery or hot spots occur in infrared imagery. The reflected component or the emitted component of the signal is usually of priority interest and generally has higher spatial frequency content. This signal is formed by the different emissivity or reflectance of each item in the image.

The detected signal is therefore a product of the illumination or background radiation and the reflectance or emissivity at each point in the image. Homomorphic filtering using spatial domain digital filters provides an effective means of dynamic range compression by providing the capability to suppress the lower frequency component of the signal (illumination or background radiation component) and enhancing the higher frequency component of the signal (reflected or emitted component of the signal) [18]. This procedure is accomplished by taking the logarithm of the input signal, filtering with a high frequency boost filter and exponentiating the resulting output.

7.2 Subjective Image Processing

A simple design procedure can be used to allow an untrained operator to design digital filters for subjective image processing. For example, a low pass or high pass filter may be specified by the cutoff frequency and the number of poles desired [3]. A high frequency enhancement filter or a low frequency boost filter may be specified by a break frequency and the magnitude of the boost. Thus, the user does not have to learn filter theory or be concerned with signal to noise considerations, etc. to design the desired filter. This is a very

valid approach for subjective image processing because decisions about the type of filter desired are usually made based upon experience. Thus the user should be provided with several options which can be implemented with a minimum of effort and without special training. Recursive digital filters are well suited for this application.

7.3 Bandwidth Optimization

If an imaging system is used in an interactive mode, digital filters can be used to effectively change the bandwidth of the imaging system to meet a particular application. Thus under low signal to noise operating conditions, the operator can decrease the bandwidth of the system in an attempt to improve his ability to discern details of an object of interest. This can be accomplished with spatial digital filters simply by changing filter coefficients. No change in hardware is required.

7.4 Interpolation

Often it is desired to change the size of an image in image presentation or display operations. This usually requires a change in the number of rows or columns of the subject image. In changing the size of the image, the sampling theorem must be considered. Artifacts in output images after the use of a simple interpolation scheme are quite often due to aliasing.

An image is usually stored in discrete form. That is, only samples of the image are available in the form of pixel elements. Thus interpolation really involves reconstructing the image to a continuous form and then resampling at the new desired intervals. The ideal interpolation algorithm would involve a reconstruction filter based upon the sampling theorem [5] and a sampling algorithm to resample the image at the desired intervals. However, it is not computationally feasible to use this approach. Therefore, it is common practice

to use a simple algorithm such as nearest neighbor, bilinear or constrained polynomial interpolation for image processing requirements. These algorithms all result in aliasing when either the number of rows or columns is decreased. If the number of rows or columns is increased, these algorithms add undesired noise to the output image which is image dependent [16].

A means of improving the results of these interpolation schemes is to use prefiltering to avoid aliasing and/or post filtering to remove undesirable additive noise. The results using this procedure can be made to be very close to the ideal reconstruction filter interpolator with the proper combination of filtering and a simple interpolation algorithm. The use of recursive digital filters which have been shown to be considerably more efficient computationally than the FFT algorithm for image processing makes this procedure feasible. For example, the bilinear interpolation algorithm can effectively be combined with an antialiasing filter when needed to give results which are very significantly improved over the use of the bilinear interpolation algorithm alone. Computationally, such a scheme would compare very favorably to a constrained polynomial interpolation algorithm and would give superior results for many images.

7.5 Image Registration, Classification and Evaluation

Image registration, classification and evaluation schemes generally do not take advantage of digital filtering. In general, relatively simple schemes are used with human interaction playing a very important role. This is partially true because of the inconvenience of using filtering with current techniques which employ the FFT algorithm and partially because the feasibility of using spatial filtering to improve image registration, classification and evaluation has not been demonstrated.

Two dimensional recursive digital filters have advantages which make them very attractive for use in exploring the feasibility of using spatial filtering to improve these procedures. The filters can be designed with only a small number of parameters specified by the user (usually no more than two parameters must be specified). The actual filtering process requires significantly fewer computations and data transfers than the FFT algorithm and image size is not constrained to power of 2. Thus, very fast turnaround can be achieved.

With very fast turnaround and with the availability of various types of filters, the exploration of the use of filtering for image registration, classification and evaluation becomes far more practical. If spatial filtering proves beneficial, then the implementation can be done with only a small sacrifice in time and without the use of a very large computer system. Thus two dimensional recursive digital filters may be very beneficial to image registration, classification and evaluation. In practice, the use of such filters may prove to be very beneficial in automating these vital procedures.

3.0 IMAGE PROCESSING FACILITIES

The Department of Electrical Engineering at A & T State University has a HP1000 Series E computer system and the University has a DEC10 computer system. Both of these computer systems were used with this program.

The HP1000 is a 16 bit minicomputer system with 32k words of core, a 14.6 megabyte disk drive and a 9 track tape drive. The core will be extended to 192k bytes and the CPU is being upgraded to series E with the RTE- IVB operating system. This upgrade will be completed by the end of February, 1981. The 9 track system can be used to transfer data from and to the DEC10 computer system. A Grinnell Model GMR-27 display image system is also available. This display can display an image with 256 rows and 512 pixels per row with 8 bit accuracy.

Plans also include additional graphics capability and a full color display system.

The DEC10 computer system is an interactive system with a 36 bit word length and double precision arithmetic capability. Thus, it can be used for stability analysis and filter synthesis and evaluation. The current DEC10 system consist of a KL-10 central processor, 512k words of memory, 2 self loading tape drives a communications controller for up to 96 asynchronous dial drives.

The Department of Electrical Engineering also has a HP2648 graphics terminal which is connected to the HP1000 computer. This graphics terminal is used for interactive stability analysis and filter synthesis.

9.0 IMAGE PROCESSING RESULTS

The lack of a hard copy output capability presents considerable difficulty with regard to including actual Landsat images or the processed results in this report. A HP9872 plotter is connected to the HP1000 computer and may be used to plot frequency contour and perspective plots of the actual filter used in the image processing examples. However, a 35-mm camera was used to photograph the Grinnell display screen to obtain the examples that follows.

Figure 1 is the frequency perspective plots of a 5 magnitude High Boost Filter with 0.2 cutoff frequency. Figure 2 is the frequency contour plot of the same filter. Figure 3 is file number three (3) of the Landsat Imagery tape received from NASA. Figure 4 shows the results of processing images with the filter of Figure 1 and then mapping between minimum and maximum logarithmically.

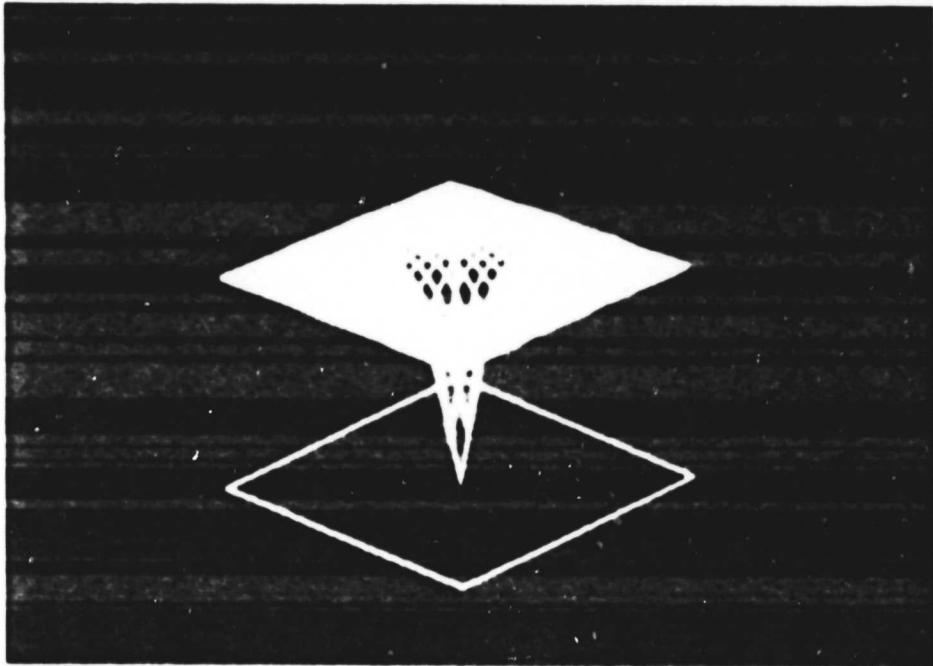


Figure 1. Perspective plot 5x-0.2 High Boost Filter

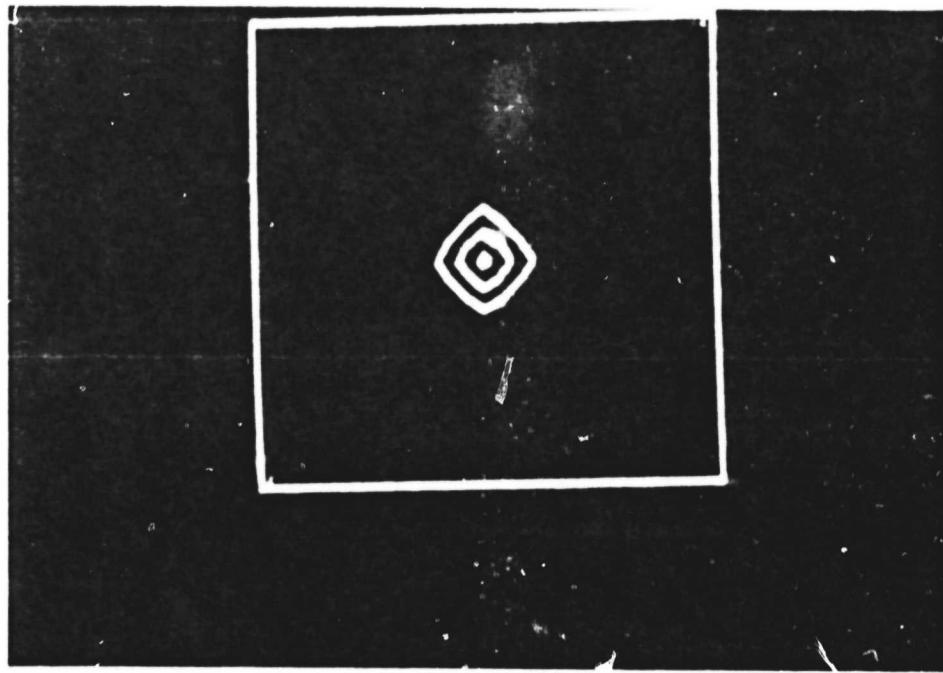


Figure 2. Contour plot 5x-0.2 High Boost Filter

ORIGINAL PAGE IS
OF POOR QUALITY

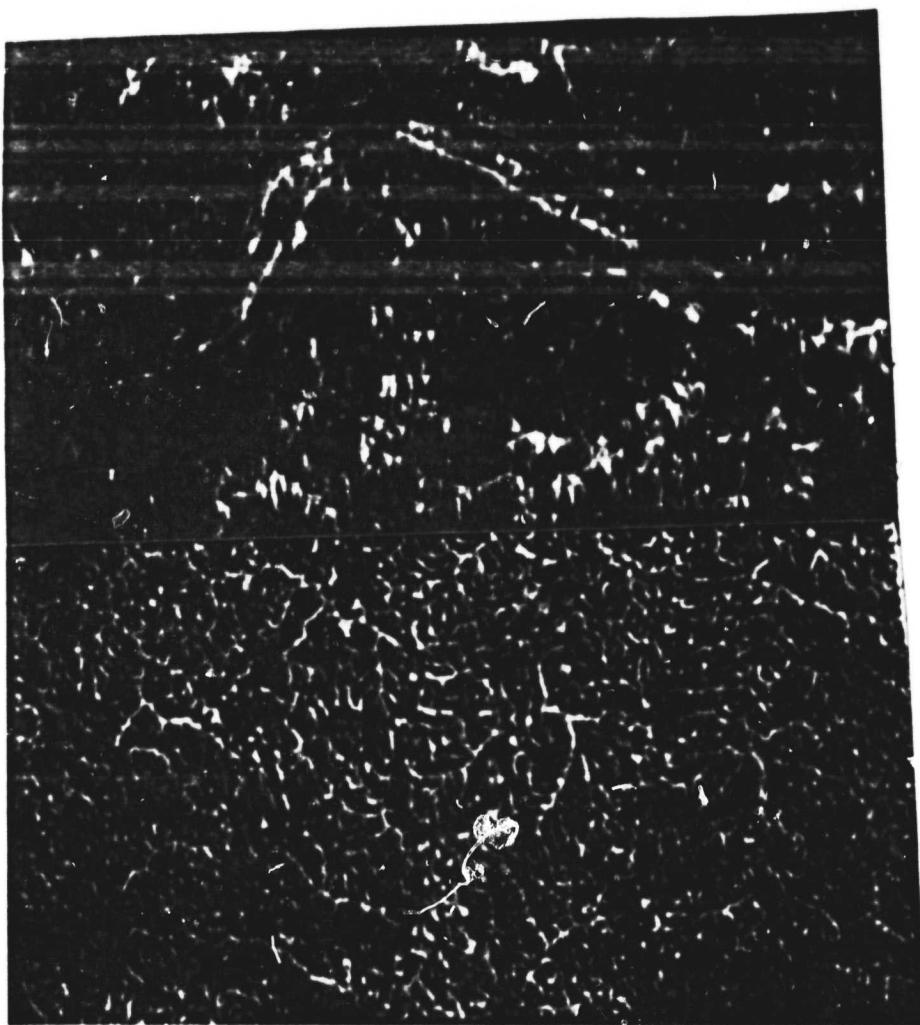


Figure 3. Original Landsat File-3 Image

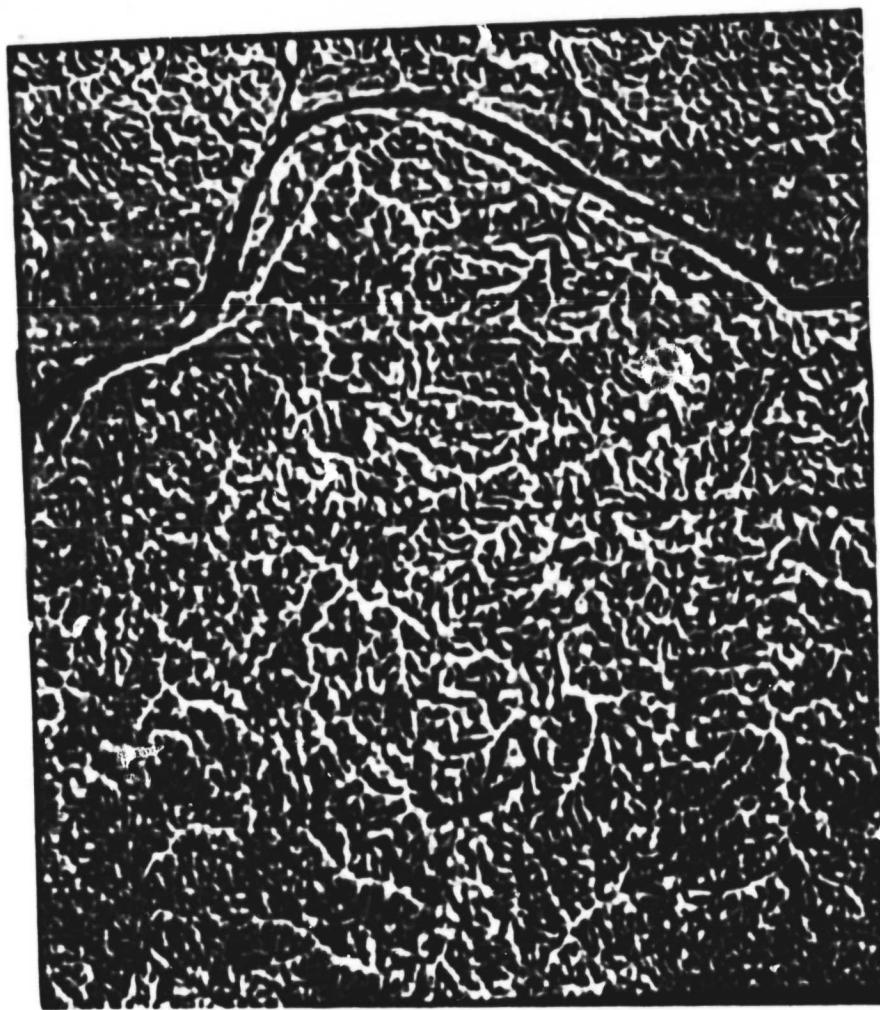


Figure 4. Enhanced Landsat file-3 Image

12.0 REFERENCES

1. Earnest L. Hall, "A Comparison of Computations for Spatial Filtering", Proceedings of the IEEE, Vol. 60, no. 7, 1972, pp 887-891.
2. Winser E. Alexander and Steven A. Pruess, "Stability Analysis of Two Dimensional Digital Recursive Filters", IEEE Transactions on Circuits and Systems, Vol. , No. 1, 1980, pp.
3. Winser E. Alexander and William J. Craft, Documentation for Spatial Domain Filtering Package, Department of Electrical Engineering, North Carolina A & T State University, January, 1979.
4. Lawrence R. Rabiner and Bernard Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1975, pp 442-455.
5. Samuel Stearns, Digital Signal Analysis, Hayden Publishing Co., Inc., Edison, N. J.
6. N. K. Bose, "Problems and Progress in Multidimensional System Theory", Proceedings of the IEEE, Vol. 65, No. 6, 1977, pp. 824-840.
7. Katsuhiko Ogata, State Space Analysis of Control Systems, Prentice-Hall, Inc., Englewood Cliffs, N. J., p. 487.
8. E. Fornasini and G. Marchesini, "State Space Realization Theore for Two Dimensional Filters", IEEE Transactions on Automatic Control, Vol. AC-21 1976, pp 484-492.
9. E. L. Hall, Digital Filtering of Images, Ph.D. Dissertation, University of Missouri, Columbia, Mo., 1971.
10. A. Papoulis, Systems and Transforms with Applications in Optics, McGraw-Hill Book Co. New York, N. Y., 1968, p. 140.
11. Winser E. Alexander, A Study of Two Dimensional Recursive Digital Filters, Final Report (Naval Air Systems Commd Contract No. NO-14-77-C-0199), School of Engineering, N. C. A T State University, Greensboro, N. C., November, 1978.
12. Winser E. Alexander and Earnest E. Sherrod, "Two Dimensional Recursive Digital Filters for Subjective Image Processing", 13th Asilomar Conference on Circuits, Systems and Computers, November, 1979.
13. J. M. Costa and A. N. Venetsonopoulos, "Design of Circularly Symmetric Two Dimensional Recursive Filters", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-22, No. 6, 1974, pp 432-442.
14. Dennis Goodman, "A Design Technique for Circularly Symmetric Low Pass Filters", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-26, No. 4, 1978, pp 290-304.

15. B. R. Hunt, "Data Structures and Computational Organization in Digital Image Enhancement", Proceedings of IEEE, Vol. 60, 1972, pp 884-887.
16. William K. Pratt, Digital Image Processing, John Wiley and Sons, Inc., Somerset, N. J., 1978.
17. Winser E. Alexander, "Electronic Target Enhancement in Infrared Reconnaissance", Proceedings of the 1968 Air Force Science and Engineering Symposium, October, 1968.
18. Thomas G. Stockham, Jr. "Image Processing in the Context of a Visual Model", Proceedings of the IEEE, Vo' 60, 1972, pp 828-842.

APPENDIX A

INVESTIGATION OF ALTERNATIVE REALIZATION TECHNIQUES

Another aspect of the research conducted under this contract was that of investigating alternative realization techniques for not only the filter designs chosen, but also for a more general class of filters as well. This investigation although as yet incomplete has resulted in some interesting conceptual reformulations of the filter realization problem [1], as well as the suggestion of possibly more computationally efficient algorithms for obtaining the filter solutions.

The typical approach taken in realizing recursive 2 D filters is one of processing the filtered output directly using the forward and backward difference equation formulations of the filter. This approach requires that one either already know the initial condition or boundary condition state of the filtered output (which generally is not the case), or that one uses various statistical estimates of what these boundary states might be in order to begin the recursion. In either case the direct use of the difference equations may not result in a minimum number of arithmetic operations being performed in obtaining a filtered solution [2,3,4].

The approach taken in this aspect of the conducted research was one of formulating the complete set of simultaneous linear algebraic equations to be solved in order to obtain a solution which satisfies the 2 D difference equation description of the filter. This serves to give one a complete description of the constraints which must be satisfied by the filtered solution with or without boundary conditions imposed on the problem.

The class of filters considered were those which possess a rational transfer function. Such a filter may be represented by its bivariate difference equation

written in tensor form as:

$$b_{ij} g_{p+i,q+j} = a_{ij} f_{p+i,q+j} \quad (1)$$

where $1 \leq p \leq N$, $1 \leq q \leq M$, $-m \leq i \leq m$, $-m \leq j \leq m$; and the double appearance of an indice on a given side of the equality implying the usual tensor notation for a summation over the specified range of that indice. The so called finite duration impulse response filter (FIR) is one which satisfies $b_{00}=1$ with all other $b_{ij}=0$; whereas the infinite duration impulse response filter (IIR) is one which allows nonzero g_{ij} for $i,j \neq 0$. A more formal tensor expression for (1) is given by:

$$B_{pq}^{k1} g_{k1} = A_{pq}^{k1} f_{k1} \quad (2)$$

where $1 \leq k \leq N$, $1 \leq l \leq M$, and the non-zero components of the coefficient tensors given by $A_{pq}^{k1} = a_{k-p,l-q}$ and $B_{pq}^{k1} = b_{k-p,l-q}$; for $-m \leq k-p \leq m$ and $-m \leq l-q \leq m$. The 2 D filtering operation requires that one determine all the elements g_{pq} , given all the coefficients a_{ij} , b_{ij} , and the input array f_{pq} .

A solution to equation (2) will exist and be unique if there exists an inverse of the tensor B_{pq}^{k1} , say C_{uv}^{pq} ; with $1 \leq u \leq N$, $1 \leq v \leq M$. For such a case, the filtered solution would then be given by:

$$g_{uv} = C_{uv}^{pq} A_{pq}^{k1} f_{k1} \quad (3)$$

Tensor equation (2) can also be interpreted as a matrix equation with A_{pq}^{k1} , and B_{pq}^{k1} taken as NM by NM dimensional coefficient matrices with row index "pq", column index "k1"; and g_{k1} and f_{k1} interpreted as column vectors. Viewing equation (2) as such a matrix equation reveals the enormity of the computer storage problem encountered in attempting a solution, for if both N

and M were typically of the order to say 512 (for a 512 by 512 pixel array) then 2^{36} memory locations would be required for the tensor of matrix B_{pq}^{k1} alone.

The matrix equation interpretation of equation (2) also reveals the following characteristics of the coefficient matrix B_{pq}^{k1} for these selected digital filters:

- (a) For the "Quarter Plane" digital filter, B_{pq}^{k1} is a triangular matrix. Hence, the solution for the filtered array g_{k1} requires no inversion of the coefficient matrix. By a simple back substitution process, starting at one corner of the array and proceeding by rows or columns, the filtered array may be computed provided that the iteration process is numerically stable.
- (b) For the "Symmetric" digital filter, with filter coefficients symmetric with respect to any diagonal passing through the central element b_{00} of the mask b_{ij} , the coefficient matrix B_{pq}^{k1} is symmetric.

Among the interesting results developed during the tenure of this research was the fact that for square arrays $N=M$, and filters with $a_{00}, b_{00} \neq 0$; the filtering problem given by equation (2) is also expressible as a matrix equation involving only N by N dimensional sparse coefficient matrices given by:

$$LGR + \sum_{k=-m, k \neq 0}^m S_k G T_k = c PFQ + c \sum_{k=-m, k \neq 0}^m S_k F U_k \quad (4)$$

where $c = a_{00}/b_{00}$, the matrix $G = (g_{pq})$ is the filtered array, $F = (f_{pq})$ is the input array; and the nonzero components of the coefficient matrices L, R, P, Q, S_k, T_k , and U_k are given by:

(i) For p, q such that $-m \leq q-p \leq m$:

$$L_{pq} = b_{q-p,0}/b_{00}; R_{pq} = b_{0,q-p}/b_{00}; P_{pq} = a_{q-p,0}/a_{00}; Q_{pq} = a_{0,q-p}/a_{00};$$

$$T_{kpq} = b_{k,p-q}/b_{00} - b_{k,0}b_{0,p-q}/b_{00}^2; U_{kpq} = a_{k,p-q}/a_{00} - a_{k,0}a_{0,p-q}/a_{00}^2.$$

(ii) And finally, for p, q such that $q-p=k$: $S_{kpq} = 1$.

The reduction in the dimensions of the coefficient matrices shown in equation (4) is one of the practical reasons why one would prefer to solve that expression for the filtered output rather than equation (2). The coefficient matrices in (4) also have other appealing properties in that both L and R are symmetric matrices, all of the matrices have the "bandtype" structure in that they have but one distinct element per respective major or minor diagonal, and all of the matrices are relatively sparse (many zero elements).

Unfortunately expression (4) is not generally solvable by using linear methods due to the fact that one cannot combine those matrices which premultiply the unknown matrix G (i.e., L and the S_k), or those matrices which postmultiply G (i.e., R and the T_k). It should be noted, however, that for those cases in which equation (4) is not solvable for G using linear methods, this does not imply that there exists no unique solution. It is equation (2) that dominates in that it is always solvable if (4) is solvable, but (2) may still be solvable even if (4) is not linearly solvable. Hence, from the standpoint of linear analysis (2) possesses more potential in solving for g_{pq} than equation (4).

There is an important class of filters for which equation (4) is linearly solvable, and this class is the set of filters which are product separable.

The coefficients involved in product separable filters have the properties:

$$a_{k,p-q}/a_{00} - a_{k,0}a_{0,p-q}/a_{00}^2 = 0$$

$$b_{k,p-q}/b_{00} - b_{k,0}b_{0,p-q}/b_{00}^2 = 0$$

Hence, the matrices T_k , and U_k are all identically zero and equation (4) reduces to:

$$LGR = c PFQ \quad (5)$$

and the solution for the filtered output G given by:

$$G = L^{-1}(cPFQ) R^{-1} \quad (6)$$

At first glance it would appear the the computation of the filtered output array G is still a formidable task due to the required inversions L^{-1} , and R^{-1} ; however both L , and R are Toeplitz matrices and can be inverted efficiently [5], hence we have our first instance of a possibly more efficient algorithm for obtaining filter solutions.

Adding additional restrictions, it has also been determined that if the filter is both product separable as well as symmetric then the coefficient matrices L and R can be further decomposed to give equation (5) the equivalent expression:

$$L_u L_1 G R_u R_1 = c PFQ \quad (7)$$

where L_1 and R_1 are lower triangular, and L_u and R_u are upper triangular matrices. Expression (5) is then solvable for G using a minimum number of arithmetic operations without requiring the inversion of L and R , provided that the intermediate results are numerically stable.

Finally, for the filter problem described by expression (4), iterative methods of solution such as:

$$G^{(n+1)} = L^{-1} \left(\sum_{k=-m, k \neq 0}^m S_k G^{(n)} T_k \right) R^{-1} + H \quad (8)$$

$$\text{where } H = L^{-1} (cPFQ + c \sum_{k=-m, k \neq 0}^m S_k F U_k) R^{-1}$$

as suggested as possible techniques to be applied to obtain filter solutions for those filters which do not satisfy the restrictions required for expressions (5), (6), and (7). The investigation of the convergence of such iterative solution techniques is the subject of current and future research.

REFERENCES

- [1]. D. E. Olson, W. E. Alexander and E. E. Sherrod, "Simultaneous Linear Algebraic Equation Formulations of Two Dimensional Digital Filter Realizations," Eighteenth Annual Allerton Conference on Communications, Control, and Computing Proceedings, October 1980.
- [2]. R. M. Mersereau, and D. E. Dudgeon, "Two-Dimensional Digital Filtering," IEEE Proc. 63(4): 610-623 (1975).
- [3]. S. K. Mitra, A. D. Sagar, and N.A. Pendergrass, "Realizations of Two Dimensional Recursive Digital Filters," IEEE Trans. Circuits Syst. CAS-22(3): 177-184 (1975).
- [4]. E. L. Hall, "A Comparison of Computations for Spatial Filtering," IEEE Proc. 65(6), June, 1977.
- [5]. S. Zchar, "Toeplitz Matrix Inversion: The Algorithm of W. F. Trench," Journal of the Assoc. for Computing Machinery, Vol. 16, No. 4, October 1969: 592-601.

Appendix B

Implementation Consideration for Two Dimensional Recursive Digital Filters with Product Separable Denominators.

Introduction

Consideration is given to the implementation of two dimensional digital recursive filters that have transfer functions with product separable denominators. This structure is of particular importance to this program because the design technique used for the design of approximately circularly symmetric filters results in a transfer function with a product separable denominator. We seek to derive a computationally efficient structure that may also lend itself to implementation with the use of a pipeline or array processor.

Transfer Function

The bivariate Z-transform for the structure of interest is given by

$$H(z,w) = \frac{\sum_{J=0}^2 \sum_{K=0}^2 a_{JK} z^{-J} w^{-K}}{\sum_{J=0}^2 \sum_{K=0}^2 b_{JK} z^{-J} w^{-K}} = \frac{N(z,w)}{D(z,w)} \quad (1)$$

We have assumed that L=2 for a single second order filter stage. We also assume that the denominator polynomial, D(z,w) can also be represented as

$$D(z,w) = \left[\sum_{J=0}^2 c_J z^{-J} \right] \left[\sum_{K=0}^2 d_K w^{-K} \right] \quad (2)$$

We can implement $H(z,w)$ in cascade form

$$H(z,w) = H_1(z,w)H_2(z,w)H_3(z,w) \quad (3)$$

where

$$H_1(z,w) = \sum_{J=0}^2 \sum_{K=0}^2 a_{JK} z^{-J} w^{-K} \quad (4)$$

$$H_2(z,w) = 1/ \sum_{J=0}^2 c_J z^{-J} \quad ; \quad c_0 = 1 \quad (5)$$

$$H_3(z,w) = 1/ \sum_{K=0}^2 d_K w^{-K} \quad ; \quad d_0 = 1 \quad (6)$$

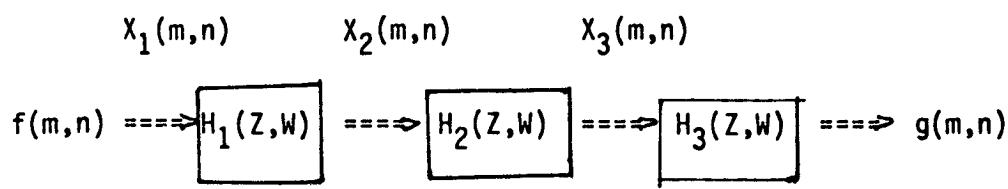
In direct form, the corresponding difference equations are given by

$$x_1(m,n) = \sum_{J=0}^2 \sum_{K=0}^2 a_{JK} f(m-J, n-K) \quad (7)$$

$$x_2(m,n) = x_1(m,n) - c_1 x_2(m-1,n) - c_2 x_2(m-2,n) \quad (8)$$

$$g(m,n) = x_2(m,n) - d_1 x_3(m,n-1) - d_2 x_3(m,n-2) \quad (9)$$

Note that this form only requires 13 multiplies and 13 adds as compared to 17 multiplies and 17 adds for the direct form associated with (1). The block diagram for this implementation is given below.



APPENDIX C

PROGRAM NAME: NASA

TYPE: Transfer

PROGRAMMER: W.E. ALEXANDER

Source:

Reloc:

FUNCTION: .nis transfer off and RP's all necessary modules
for Image Processing; mounts cartridge 23 and runs NASA 1.

FROM RTE: Run NASA

Modules Called:

SHOW
BLDWF
BLDIM
WTAPE
DSPLY
CURSR
FDIGN
STABI
DPLAM
FILTR
LFLTR
HFLTR
RESIZ
IMAGE
DINTP
NOISE
FIRO

Modules Run: NASA1

Subroutines Called:

PROGRAM NAME: NASA1

TYPE: Program

PROGRAMMER: W.E. ALEXANDER

SOURCE: &NASA1

Reloc: %NASA1

FUNCTION: This Program is the father program for the Image Processing from which the major modules are selected.

Modules Called:

DSPLY
FDIGN
FILTR
RESIZ
SHOW
BLDIM
NOISE

Modules Run:

Subroutines Called:

FILL

PROGRAM NAME: DSPLY

TYPE: Program

PROGRAMMER: DAVE JOHNSON

Source: &DSPLY

Reloc: %DSPLY

FUNCTION: This program Displays an Image on the Grinnell Image
Display System GMR-27.

Modules Called:

SCROL
CURSR

Modules Run:

Subroutines Called:

WLINE
RLINE
DRIVR
RESET
MOVEC

PROGRAM NAME: FDIGN

TYPE: Program

PROGRAMMER: E.E. SHERRUD

Source: &FDIGN, &FDIG1

Reloc: %FDIGN, %FDIG1

FUNCTION: This program designs, stability tests and displays a filter on either HP-2648G or on the Grinnell GMR-27.

Modules Called:

STABI
DPLAM
FIRO
PLOTV

Data File Created:

COEFFS
DATA1

Subroutines Called:

LPFLT
BPFLT
BSTFT
TDLPF
ROTAE
FIR

PROGRAM NAME: STABI

TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &STABI

Reloc: %STABI

FUNCTION: This Program evaluates the Recursive Filter Stability
Characteristics.

Modules Called:

Subroutines Called:

STABT
PRTN

PROGRAM NAME: DPLAM

TYPE: Program

Source: &DPLAM, &DPLA1

Reloc: %DPLAM, %DPLA1

PROGRAMMER: E.E. SHERROD

FUNCTION: This program displays the Filter Characteristics.

Modules Called:

COEFFS
DPLA1

Subroutines Called:

ZWC
CONTR
SET3D
PLT3D
SET2U
PLT2D

PROGRAM NAME: FIRO

TYPE: PROGRAM

PROGRAMMER: E.E. SHERROD

Reloc: %FIRO

Source: &FIRO

FUNCTION: This program designs Non-Recursive FIR Filters.

Modules Called:

Subroutines Called:

BESJ

BESIO

PROGRAM NAME: PLOTV

TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &EES3

Reloc: %PLOTV

FUNCTION: This program displays Filter Characteristics on the
Grinnell Display GMR-27.

Modules Called:

DATA1

Subroutines Called:

DVECT

PROGRAM NAME: FILTR

TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &FILTR

Reloc: %FILTR

FUNCTION: This program schedules Linear or Homomorphic
filtering of Images.

Modules Called:

LFLTR

HFLTR

SHOW

BLDWF

Subroutines Called:

PROGRAM NAME: BLDWF

TYPE: Program

PROGRAMMER: DAVE JOHNSON

Source: &BLDWF

Reloc: %BLDWF

FUNCTION: This program creates and maintains an Image
work file named WF0000 with pixel values stored
as 15-bit real numbers.

Modules Called:

DIREC
WF0000

Subroutines Called:

ICMPW

PROGRAM NAME: LFLTR

TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &LFLTR

Reloc: %FILTR

FUNCTION: This program does Linear Filtering using Spatial Domain
Recursive Digital Filters.

Modules Called:

COEFFS

Subroutines Called:

READL
RITLN
FILTR
WFINT
CLSWF

PROGRAM NAME: HFLTR

TYPE: Program

PROGRAMMER: E.E. SHERROD

Source: &HFLTR

Reloc: %HFLTR

FUNCTION: This program performs Homomorphic Filtering using
Spatial Domain Recursive Digital Filters.

Modules Called:

COEFS

Subroutines Called:

WFINT

READL

RITLN

HFILT

CLSWF

PROGRAM NAME: RESIZ

TYPE: Program

PROGRAMMER: W.E. ALEXANDER and RICHARD MOORE

Source: &RESIZ

R ELOC: %RESIZ

FUNCTION: This program allows the user to scale an Image and change an Image from 8-bits to 15-bits and vice versa. The resizing of an Image is being developed.

Modules Called:

LFLTR
DINTP
TRMGN
LBRSZ
BLDWF

Subroutines Called:

TRMGN
BLANX
SPCHR
CKFLD
WFINT
READL
XYFLT
CLSWF
READL
RITEL

PROGRAM NAME: SHOW

TYPE: Program

PROGRAMMER: DAVE JOHNSON

Source: &SHOW

Reloc: %SHOW

FUNCTION: This program displays an image from the work
file onto the Grinnell System GMR-27.

Modules Called:

WF0000

Subroutines Called:

READL

WLINE

CLSWF

PROGRAM NAME: BLDIM

TYPE: Program

PROGRAMMER: DAVE JOHNSON

Source: &BLDIM

Reloc: %BLDIM

Loadfile: LBLDIM

FUNCTION: This program constructs an 8 or 15-bit image from
magnetic tape, disc, GMR-27 display or work file.

Modules Called:

WF0000
DIREC

Subroutines Called:

MVW
ROT8
DCODE
DRIVR

PROGRAM NAME: NOISE

TYPE: Program

PROGRAMMER: E.E SHERROD

Source: &NOISE

Reloc: %NOISE

FUNCTION: This program add Gaussian Noise to an Image with user defined Mean and Standard Deviation from a Gaussian Noise disc file.

Modules Called:

BLDWF

Subroutines Called:

READL

RITEL

CLSWF

.

APPENDIX D

LOAD FILES

Table of Content:

Page 1

LDIREC-----	1
LWTAPE-----	1
LDPLAM-----	1
LPLOTV-----	1
LFDIGN-----	2
LBLDIM-----	2
LLFTR-----	2
LHFTR-----	3
LSHOW-----	3
LFIRO-----	3
LIMAGE-----	4
LRESIZ-----	4
LDINTP-----	4
LBLDWF-----	5
LDSPLY-----	5
LCURSR-----	5
NASA-----	6

PROGRAM FILES

Table of Content:

Page ii

&NASA1-----	7
&DSPLY-----	9
&CURSR-----	12
&ICMPW-----	13
&RESET-----	13
&RLINE-----	14
&MOVEC-----	15
&IMAGE-----	16
&SPAGE-----	23
&RESIZ-----	24
&ROT8-----	33
&TRMGN-----	34
&LBRSZ-----	37
&WFINT-----	39
&DINTP-----	42
&WTAPE-----	44
&EES3-----	47
&DPLAM-----	48
&DPLA1-----	52
&FDIGN-----	60
&BLDIM-----	70
&LFLTR-----	76
&HFLTR-----	82
&SHOW-----	88
&FIRO-----	89
&WINDO-----	92
&BESIO-----	92
&BESJ-----	93
&BLDWF-----	95
&SCROL-----	98
&WLINE-----	99
&DRIVE-----	100
&FD1G1-----	102
&STABI-----	105
&FILTR-----	117

LDIREC T=00004 IS ON CR00025 USING 00002 BLKS R=0002

0001 :PU,IMDIRC:IM: 23:4:100
0002 :^R,IMDIRC:IM: 23:4:100

LWTAPE T=00004 IS ON CR00022 USING 00002 BLKS R=0010

0001 :LG,3
0002 :SV,0
0003 :OF,WTAPE
0004 :PU,WTAPE
0005 :MR,%WTAPE
0006 :MR,%ZROT8
0007 :MR,%ICMPW
0008 :MR,%ZTPMGN
0009 :RU,LOADR,99,0G,,,2
0010 :SP,10G::3
0011 :TR

LDPLAM T=00004 IS ON CR00022 USING 00002 BLKS R=0010

0001 :SV,0
0002 :OF,DPLAM
0003 :PU,DPLAM
0004 :LG,3
0005 :MR,%DPLAM
0006 :MR,%DPLA1
0007 :RU,LOADR,99,0G
0008 :SP,10G::3
0009 :OF,10G
0010 :RP,10G::3
0011 ::

LPLOTV T=00004 IS ON CR00022 USING 00002 BLKS R=0011

0001 :SV,0
0002 :OF,PLOTV
0003 :PU,PLOTV
0004 :LG,3
0005 :MR,%EES3
0006 :MR,%DRIVR
0007 :RU,LOADR,99,0G
0008 :SP,10G::3
0009 :OF,10G
0010 :RP,10G::3
0011 ::

LFDIGN T=00004 IS ON CR00022 USING 00002 BLKS R=0011

```
0001 :SV,0
0002 :OF,FDIGN
0003 :PU,FDIGN
0004 :LG,3
0005 :MR,ZFDIGN
0006 :MR,ZFDIG1
0007 :RU,LOADR,99,1G
0008 :PU,10G::3
0009 :SP,10G::3
0010 :OF,10G
0011 :::
0012
```

LBLDIM T=00004 IS ON CR00022 USING 00002 BLKS R=0009

```
0001 :LG,2
0002 :OF,BLDIM
0003 :PU,BLDIM
0004 :MR,ZBLDIM
0005 :MR,MVW.
0006 :MR,ZROTS
0007 :MR,ZRLINE
0008 :MR,DCODE.
0009 :MR,ZDRIVR
0010 :RU,LOADR,99,0G,,,2
0011 :SP,10G::3
0012 :OF,10G::3
0013 :RP,10G::3
0014 :::
```

LLFTR T=00004 IS ON CR00022 USING 00002 BLKS R=0011

```
0001 :SV,0
0002 :OF,LFLTR
0003 :PU,LFLTR::3
0004 :LG,3
0005 :MR,ZLFLTR
0006 :MR,ZWFINT
0007 :MR,DCODE.
0008 :MR,ZWLINE
0009 :MR,ZDRIVR
0010 :RU,LOADR,99,1G
0011 :SP,10G::3
0012 :OF,10G
0013 :RP,10G::3
0014 :::
```

LHFTR T=00004 IS ON CR00022 USING 00002 BLKS R=0014

```
0001 :SV,0
0002 :OF,HFLTR
0003 :PU,HFLTR::3
0004 :LG,3
0005 :MR,ZHFLTR
0006 :MR,ZWFINT
0007 :RU,LOADR,99,1G
0008 :SP,10G::3
0009 :OF,10G
0010 :RP,10G::3
0011 ::
```

LSHOW T=00004 IS ON CR00022 USING 00002 BLKS R=0011

```
0001 :LG,1
0002 :MR,ZSHOW
0003 :MR,ZWFINT
0004 :MR,ZDRIVR
0005 :MR,ZWLINE
0006 :OF,SHOW
0007 :RU,LOADR,99,1G
0008 :PU,10G::3
0009 :SP,10G::3
0010 :OF,10G
0011 ::
```

LFIRO T=00004 IS ON CR00022 USING 00002 BLKS R=0010

```
0001 :LG,1
0002 :MR,ZFIRO
0003 :MR,ZWINDO
0004 :MR,ZBESIO
0005 :OF,FIRO
0006 :RU,LOADR,99,0G
0007 :PU,10G::3
0008 :SP,10G::3
0009 :OF,10G
0010 :RP,10G::3
0011 ::
```

LIMAGE T=00004 IS ON CRO0022 USING 00002 BLKS R=0009

```
0001 :LG,1
0002 :OF,IMAGE
0003 :MR,%IMAGE
0004 :MR,%SPACE
0005 :MR,%ICMPW
0006 :MR,%ROTS
0007 :RU,LOADR,99,1G
0008 :PU,10G::3
0009 :SP,10G::3
0010 :OF,10G
```

LRESIZ T=00004 IS ON CRO0022 USING 00002 BLKS R=0004

```
0001 :LG,3
0002 :SV,0
0003 :OF,RESIZ
0004 :PU,RESIZ
0005 :MR,%RESIZ
0006 :MR,%DRIVR
0007 :MR,%WLINE
0008 :MR,%TRMGN
0009 :MR,%LBRSZ
0010 :MR,%WFINT
0011 :RU,LOADR,99,0G,,,2
0012 :SP,10G::3
0013 :TR
```

LDINTP T=00004 IS ON CRO0022 USING 00002 BLKS R=0011

```
0001 :LG,3
0002 :SV,0
0003 :OF,DINTP
0004 :PU,DINTP
0005 :MR,%DINTP
0006 :MR,%DRIVR
0007 :MR,%WLINE
0008 :MR,%TRMGN
0009 :MR,%LBRSZ
0010 :MR,%WFINT
0011 :RU,LOADR,99,0G,,,2
0012 :SP,10G::3
0013 :TR
```

LBLDWF T=00004 IS ON CR00022 USING 00002 BLKS R=0009

```
0001 :LG,0
0002 :LG,2
0003 :OF,BLDWF
0004 :PU,BLDWF
0005 :MR,%BLDWF
0006 :MR,%ICMPW
0007 :RU,LOADR,99,0G,,,2
0008 :SP,10G::3
0009 :OF,10G::3
0010 :RP,10G::3
0011 ::
```

LDSPLY T=00004 IS ON CR00022 USING 00002 BLKS R=0011

```
0001 :LG,1
0002 :MR,%DSPLY
0003 :MR,%SCROL
0004 :MR,%WLINE
0005 :MK,%DRIVR
0006 :MR,%RESET
0007 :OF,DSPLY
0008 :RU,LOADR,99,0G
0009 :PU,10G::3
0010 :SP,10G::3
0011 :OF,10G
```

LCURSR T=00004 IS ON CR00022 USING 00002 BLKS R=0012

```
0001 :LG,1
0002 :MR,%CURSR
0003 :MR,%WLINE
0004 :MR,%RLINE
0005 :MR,%DRIVR
0006 :MR,%MOVEC
0007 :OF,CURSR
0008 :RU,LOADR,99,1G
0009 :PU,10G::3
0010 :SP,10G::3
0011 :OF,10G
0012 ::
```

LNOISE T=00004 IS ON CR00022 USING 00002 BLKS R=0011

0001 :LG,1
0002 :MR,%NOISE
0003 :MR,%BLDWF
0004 :MR,%WFINT
0005 :MR,%ICMPW
0006 :OF,NOISE
0007 :RU,LOADR,99,1G
0008 :PU,10G::3
0009 :SP,10G::3
0010 :OF,10G
0011 ::

NASA T=00004 IS ON CR00022 USING 00002 BLKS R=0008

0001 :SV,4
0002 :OF,SHOW
0003 :RP,SHOW
0004 :OF,BLDWF
0005 :RP,BLDWF
0006 :OF,BLDIM
0007 :RP,BLDIM
0008 :OF,WTAPE
0009 :RP,WTAPE
0010 :OF,DSPLY
0011 :RP,DSPLY
0012 :OF,CURSR
0013 :RP,CURSR
0014 :OF,FDIGN
0015 :RP,FDIGN
0016 :OF,STABI
0017 :RP,STABI
0018 :OF,DPLAM
0019 :RP,DPLAM
0020 :OF,FILTR
0021 :RP,FILTR
0022 :OF,LFLTR
0023 :RP,LFLTR
0024 :OF,PLOTV
0025 :RP,PLOTV
0026 :OF,HFLTR
0027 :RP,HFLTR
0028 :OF,RESIZ
0029 :RP,RESIZ
0030 :OF,IMAGE
0031 :RP,IMAGE
0032 :OF,DINTP
0033 :RP,DINTP
0034 :OF,NOISE
0035 :RP,NOISE
0036 :MC,23
0037 :SV,0
0038 :RU,NASA1
0039 :OF,SHOW
0040 :OF,BLDWF
0041 :OF,DSPLY
0042 :OF,CURSR
0043 :OF,FDIGN
0044 :OF,STABI
0045 :OF,DPLAM
0046 :OF,FILTR
0047 :OF,LFLTR
0048 :OF,PLOTV
0049 :OF,HFLTR
0050 :OF,RESIZ
0051 :OF,IMAGE
0052 :OF,BLDIM
0053 :OF,DINTP
0054 :OF,WTAPE
0055

SNASA1 T-00004 IS ON CRO0022 USING 00008 BLKS R-0054

```

0001  FTN4,L
0002      PROGRAM NASA1
0003  C      THIS PROGRAM IS THE FATHER PROGRAM FOR THE IMAGE FILTERING
0004  C      PROGRAMS
0005  C
0006      DIMENSION IPRAM(5),NAME(3),NSON(3,8),IMESS(30)
0007      DATA NSON/2HDS,2HPL,2HY ,2HFD,2HIG,2HN ,2HFI,2HLT,
0008      *2HR ,2HRE,2HSI,2HZ ,2HSH,2HOW,2H ,2HIM,2HAG,2HE ,
0009      *2HNO,2HIS,2HE /
0010  C
0011  C      SON PROGRAM NAMES (FILES SAME PRESEDED WITH "&")
0012  C          DSPLY - DISPLAY PROGRAM
0013  C          FDIGN - FILTER DESIGN MODULE
0014  C          FILTR - FILTER IMPLEMENTION MODULE
0015  C          RESIZ - IMAGE MODIFICATION MODULE
0016  C          SHOW - DISPLAYS WORK FILE
0017  C          IMAGE - IMAGE DATA MANAGEMENT MODULE
0018  C          NOISE - ADDITIVE GAUSSIAN NOISE
0019  C
0020      CALL RMPAR(IPRAM)
0021      LU=IPRAM(1)
0022      IF(LU.LE.0) LU=1
0023  C
0024  C      NPROG IS THE NUMBER OF SONS
0025  C
0026      NPROG=6
0027      ICNT=9
0028  C
0029  C      DISPLAY MENU
0030  C
0031      5 WRITE(LU,30)
0032  30 FORMAT(" SELECT PROCESSING OPTION"/," 1. IMAGE DISPLAY"/,
0033      *FILTER DESIGN"/," 3. FILTER IMAGE"/," 4. MODIFY IMAGE"/,
0034      * SHOW WORK FILE"/," 6. IMAGE DATA MANAGEMENT"/," 7. NOIS
0035      *ON"/," 8. TERMINATE PROGRAM")
0036      READ(LU,*) IOPT
0037      IF(IOPT.EQ.0.OR.IOPT.EQ.1) IOPT = 1
0038      IF(IOPT.LT.1.OR.IOPT.GT.8) GO TO 16
0039      IF(IOPT.EQ.8) GO TO 500
0040  C
0041      IPARAM(2)=IOPT
0042      DO 10 I=1,3
0043  10 NAME(I)=NSON(I,IOPT)
0044      WRITE(LU,15) NAME
0045  15 FORMAT(" MODULE TO BE SCHEDULED IS ",3A2)
0046      GO TO 20
0047  16 WRITE(LU,17)
0048  17 FORMAT(" INVALID RESPONSE")
0049      GO TO 5
0050  C

```

```
0051 C
0052 20 ICNW=LU+200B
0053     CALL EXEC(13,ICNW,IPRAM(3),IPRAM(4),IPRAM(5))
0054     CALL EXEC(23,NAME,IPRAM(1),IPRAM(2),IPRAM(3),IPRAM(4),IPRAM(
0055     WRITE(LU,40) (IPRAM(I),I=1,5)
0056     40 FORMAT("PARAMETERS RETURNED FROM MODULE"/,5(1H,4E11.3,2X))
0057     GO TO 5
0058 500 CONTINUE
0059 C
0060 C      OF ALL SON PROGRAMS
0061 C
0062     DO 510 I=1,NPRG
0063     CALL FILL(IMESS,2H ,30)
0064     CALL CODE
0065     WRITE(IMESS,520) (NSON(J,I),J=1,3)
0066 520 FORMAT("OF,",3A2)
0067     IRTN=MESSS(IMESS,ICNT,LU)
0068     IF(IRTN.LT.0) CALL EXEC(2,LU,IMESS,IRTN)
0069 510 CONTINUE
0070 C
0071     STOP
0072     END
0073 C
0074 C
0075     SUBROUTINE FILI(IARAY,IA,N)
0076 C
0077 C      THIS SUBROUTINE FILLS ARRAY IARAY WHICH HAS N WORDS WITH THE
0078 C      OF IA.
0079 C
0080     DIMENSION IARAY(N)
0081     DO 10 I=1,N
0082 10 IARAY(I)=IA
0083     RETURN
0084
0085 $     END
```

```
0001  FTN4
0002      PROGRAM DSPLY
0003  C
0004  C  THIS PROGRAM DISPLAYS AN IMAGE ON THE CMR-27.  IMAGE FILE MUST
0005  C  BE IN FORMAT DESCRIBED BY IMAGE DISPLAY SUBSYSTEM.
0006  C
0007      INTEGER SLU11,STRTL,STRTP,SCROL
0008  C
0009      DIMENSION NAME(6),IDCB(144),IBLK(513),ISET(10),LU(5),JNAME(3
0010      INTEGER TEXT1(38),TEXT2(38),TEXT3(38)
0011  C
0012      EQUIVALENCE (IBLK(7),IBLK7),(IBLK(8),IBLK8),(IBLK(12),IBLK12)
0013      1 (IBLK(13),JNAME),(TEXT1,IBLK(129)),(IBLK(169),TEXT2),
0014      2 (IBLK(209),TEXT3)
0015      EQUIVALENCE (ISET(5),ISET5),(ISET(6),ISET6),(ISET(7),ISET7),
0016      1 (ISET(8),ISET8),(ISET(9),ISET9)
0017  C
0018      DATA ISET/100377B,10377B,24001B,30000B,5*-1,260C2B/
0019      DATA SLU11/34011B/
0020      DATA LLA0,LEAO,LECO,LLB1,LLBX,LEB1,LEBX/64000B,44000B,54000B
0021      1 70001B,71777B,50001B,51777B/
0022  C
0023  C
0024  C  GET INPUT PARAMETERS
0025  C
0026      CALL RMPAR(LU)
0027      IF (LU .LE. 0) LU = 1
0028  C
0029  C  OPEN IMAGE DIRECTORY FILE
0030  C
0031  100  CALL OPEN(IDCB,IERR,6HIMDIRC)
0032      IF (IERR .LT. 0) GO TO 991
0033  C
0034  C  GET IMAGE FILE NAME
0035  C
0036      CALL RESET(LU)
0037      WRITE(LU,20)
0038  C
0039      WRITE(LU,21)
0040      WRITE(LU,22)
0041      WRITE(LU,23)
0042      WRITE(LU,24)
0043      WRITE(LU,26)
0044      WRITE(LU,26)
0045      WRITE(LU,26)
0046  20      FORMAT(20X,"I M A G E   D I S P L A Y   S Y S T E M//")
0047  21      FORMAT("IMAGE NAME: dB      d@/")
0048  22      FORMAT("# LINES:   dB      d@",20X,
0049      1 "# PIXELS/LINE: dB      d@/")
0050  23      FORMAT("MIN PIXEL:   dB      d@",18X,
0051      1 "MAX PIXEL:   dB      d@/")
0052  24      FORMAT("TEXT: ,/")
0053  26      FORMAT("dB",38"  ", "d@")
0054  25      FORMAT(" ")
0055  27      FORMAT(" ")
0056  105     WRITE(LU,25)
0057      READ(LU,2) NAME
0058  2      FORMAT(6A2)
0059      IF (NAME .EQ. 2H/E) GO TO 9000
```

```
0060 C
0061 C FIND IMAGE FILE
0062 C
0063     CALL RWNDF(IDCB)
0064 110     CALL READF(IDCB,IERR,IBLK,256,LEN)
0065     IF (IERR .LT. 0) GO TO 991
0066     IF (LEN .EQ. -1) GO TO 800
0067 C
0068     DO 120 I=1,6
0069     IF (IBLK(I) .NE. NAME(I)) GO TO 110
0070 120     CONTINUE
0071 C
0072 C IMAGE FOUND--CHECK IF ON DISC
0073 C
0074     IF (IBLK12 .EQ. 1) GO TO 130
0075 C
0076 C IMAGE NOT ON DISC
0077 C
0078     WRITE(LU,12)
0079 12     FORMAT(""
0080     GO TO 105
0081 C
0082 C IMAGE IS ON DISC
0083 C
0084 130     CALL CLOSE(IDCB)
0085     RMIN = IBLK(9)
0086     RMAX = IBLK(10)
0087     WRITE(LU,29)(IBLK(I),I=7,10)
0088 28     FORMAT(""
0089     CALL EXEC(2,LU,TEXT1,37)
0090     CALL EXEC(2,LU,TEXT2,37)
0091     CALL EXEC(2,LU,TEXT3,37)
0092     WRITE(LU,27)
0093     CALL OPEN(IDCB,IERR,JNAME)
0094     IF (IERR .LT. 0) GO TO 991
0095 C
0096 C EXTRACT DISPLAY INFORMATION
0097 C
0098     NUML = IBLK7
0099     NUMP = IBLK8
0100     STRTL = (256-MINO(256,NUML))/2
0101     STRTP = (512-MINO(512, NUMP))/2
0102 C
0103 500     ISET5 = IOR(LLAO,IAND(STRTL,1777B))
0104     ISET6 = IOR(LEAO,IAND(STRTP,1777B))
0105     ISET7 = LLB1
0106     ISET8 = LEB1
0107     ISET9 = IOR(LECO,IAND(STRTP,1777B))
0108 C
0109     CALL DRIVR(2,ISET,10)
0110 C
```

```
0111      IERR = 0
0112      DO 600 I=1,MIN0(NUML,256)
0113      IF (IERR .LT. 0) GO TO 991
0114      CALL READF(IDCB,IERR,IBLK,512,NUM)
0115      IF (NUM .LT. 0) GO TO 600
0116      C
0117      DO 595 J=1,NUM
0118      IBLK(J) = (255./(RMAX-RMIN))*(FLOAT(IBLK(J))-RMIN)
0119      IF (IBLK(J) .LT. 0) IBLK(J) = 0
0120      IF (IBLK(J) .GT. 377B) IBLK(J) = 377B
0121      595  CONTINUE
0122      C
0123      IBLK(NUM+1) = SLU11
0124      CALL DRIVR(40002B,IBLK,NUM+1)
0125      600  CONTINUE
0126      IFRST = 0
0127      ILAST = 255
0128      C
0129      C
0130      C OUTPUT SOFT KEY FUNCTIONS
0131
0132      WRITE(LU,29)
0133      29   FORMAT(/"FUNCTION KEYS: /")
0134      WRITE(LU,30)
0135      WRITE(LU,30)
0136      30   FORMAT(4("dB          d@      ")/)
0137      605  WRITE(LU,31)
0138      31   FORMAT(")
0139      WRITE(LU,32)
0140      32   FORMAT(" << SCROLL   SCROLL >>          CURSOR  ",)
0141      124X," NEW IMAGE      EXIT  ")
0142      610  CALL EXEC(1,LU,INPT,1)
0143      INPT = INPT-7023
0144      IF (INPT .LT. 1 .OR. INPT .GT. 8) GO TO 610
0145      C
0146      C BRANCH TO APPROPRIATE SECTION
0147      C
0148      C
0149      GO TO (1000,2000,3000,4000,5000,6000,100,9000),INPT
0150      C
0151      C
0152      C
0153      C SCROLL IMAGE BACK
0154      C
0155      1000  IERR = SCROL(IDCB,-9,NUML,IFRST,ILAST,RMAX,RMIN)
0156      IF (IERR .LT. 0) GO TO 991
0157      GO TO 610
```

```
0158 C
0159 C SCROLL FORWARD
0160 C
0161 2000 IERR = SCROL(IDCDB,17,NUML,IFRST,ILAST,RMAX,RMIN)
0162 IF (IERR .LT. 0) GO TO 991
0163 GO TO 610
0164 3000 CONTINUE
0165 C
0166 C POSITION CURSOR
0167 C
0168 4000 CALL EXEC(23,6HCURSR ,LU)
0169 GO TO 605
0170 5000 CONTINUE
0171 6000 CONTINUE
0172 GO TO 610
0173 C
0174 C TERMINATE
0175 C
0176 9000 CALL CLOSE(IDCDB)
0177 CALL RESET(LU)
0178 WRITE(LU,33)
0179 33 FORMAT("END PROGRAM")
0180 CALL EXEC(6)
0181 C
0182 C FILE NOT FOUND
0183 C
0184 800 WRITE(LU,3)
0185 3 FORMAT(""
0186 GO TO 105
0187 C
0188 991 CALL RESET(LU)
0189 WRITE(LU,9) IERR
0190 9 FORMAT("FILE ERROR",I6)
0191 CALL CLOSE(IDCDB)
0192 END
0193 $
```

SCURSR T=00004 IS ON CR00022 USING 00005 BLKS R=0037

```
0001  FTN4
0002      PROGRAM CURSR
0003  C
0004      DIMENSION LU(5),IBUF(2352),IZERO(2)
0005  C
0006      INTEGER EA,LA
0007  C
0008      DATA IZERO/44000B,64000B/
0009  C
0010  C
0011      CALL RMPAR(LU)
0012  C
0013  C
0014  C  SAVE IMAGE LINES
0015  C
0016  C
0017      DO 50 I=0,20
0018      CALL RIINE(I,0,111,IBUF(112*I+1))
0019  50      CONTINUE
0020      WRITE(LU,1)
0021  1      FORMAT("")
0022      WRITE(LU,2)
0023  2      FORMAT("      LEFT      UP      RIGHT      ",
0024      "112X,"      DOWN      ",12X,"      RETURN      ")
0025      CALL MOVEC(0,255)
0026      EA = 0
0027      LA = 255
0028  100     CALL EXEC(1,LU,INPT,1)
0029     INPT = INPT-7023
0030     IF (INPT .LT. 1 .OR. INPT .GT. 8) GO TO 100
0031  C
0032  C
0033  C  BRANCH TO APPROPRIATE SECTION
0034  C
0035  C
0036      GO TO (400,200,500,100,100,300,100,600), INPT
0037  C
0038  C  MOVE CURSOR UP
0039  C
0040  200     LA = MOD(LA+11,256)
0041     CALL MOVEC(EA,LA)
0042     GO TO 100
0043  C
0044  C  MOVE CURSOR DOWN
0045  C
0046  300     LA = MOD(LA+249,256)
0047     CALL MOVEC(EA,LA)
0048     GO TO 100
0049  C
0050  C  MOVE CURSOR LEFT
0051  C
0052  400     EA = MOD(EA+499,512)
0053     CALL MOVEC(EA,LA)
0054     GO TO 100
0055  C
```

```

0056 C MOVE CURSOR RIGHT
0057 C
0058 500 EA = MOD(EA+17,512)
0059     CALL MOVEC(EA,LA)
0060     GO TO 100
0061 C
0062 C RETURN TO PREVIOUS SCREEN
0063 C
0064 600 DO 610 I=0,20
0065     CALL WLINE(I,0,111,IBUF(112*I+1))
0066 610 CONTINUE
0067 C
0068     CALL DRIVR(2,IZERO,2)
0069 C
0070     END
0071 $

```

&ICMPW T=00004 IS ON CRO0022 USING 00002 BLKS R=0011

```

0001 FTN4
0002     FUNCTION ICMPW(IBUF1,IBUF2,ILEN)
0003     DIMENSION IBUF1(1),IBUF2(1)
0004     DO 100 I=1,ILEN
0005     IF (IBUF1(I)  NE. IBUF2(I)) GO TO 200
0006 100  CONTINUE
0007     ICMPW = 0
0008     RETURN
0009 200  ICMPW = I
0010     END
0011 $

```

&RESET T=00004 IS ON CRO0022 USING 00002 BLKS R=0017

```

0001 FTN4
0002     SUBROUTINE RESET(LU)
0003 C
0004 C
0005     WRITE(LU,1)
0006 1     FORMAT(""
0007 C
0008 C     WAIT 200 MSEC
0009 C
0010     CALL EXEC(12,0,1,0,-20)
0011 C
0012 C     CLEAR DISPLAY
0013 C
0014     WRITE(LU,2)
0015 2     FORMAT("")"
0016     END
0017 $

```

6RLINE T=00004 IS ON CR00022 USING 00005 BLKS R=0039

```
0001  FTN4,L
0002      SUBROUTINE RLINE(LINE,IPIX,JPIX,IData)
0003  C
0004  C  THIS SUBROUTINE READS A LINE FROM GMR-27.
0005  C
0006  C      WHERE
0007  C          LINE = LINE # TO READ
0008  C          IPIX = STARTING PIXEL
0009  C          JPIX = ENDING PIXEL
0010  C
0011  C          IData = BUFFER IN WHICH DATA IS RETURNED (1 PIXEL/WORD
0012  C
0013  C
0014  C      DIMENSION IData(512),INIT(5)
0015  C
0016  C      EQUIVALENCE (LLA,INIT(2)),(LEA,INIT(3)),(LEB,INIT(4))
0017  C
0018  C      DATA INIT/100377B,64000B,44000B,50000B,26002B/
0019  C
0020  C      COMPUTE DIRECTION
0021  C
0022  C          IDIRC = 1
0023  C          IF (IPIX .GT. JPIX) IDIRC = -1
0024  C
0025  C      SET UP FOR READ BACK
0026  C
0027  C          LLA = 64000B + IAND(LINE,377B)
0028  C          LEA = 44000B + IAND(IPIX,777B)
0029  C          LEB = 50000B + IDIRC + 512
0030  C          CALL DRIVR(2,INIT,5)
0031  C
0032  C      READ BACK LINE
0033  C
0034  C          NUM = IDIRC*(JPIX-IPIX)+1
0035  C          CALL DRIVR(1,IData,NUM)
0036  C
0037  C      RETURN
0038  C
0039  $
```

SMOVEC T=00004 IS ON CRO0022 USING 00004 BLKS R=0031

```
0001  FTN4,L
0002      SUBROUTINE MOVEC(IX,IY)
0003  C
0004  C  THIS SUBROUTINE MOVES THE CURSOR ON THE GMR-27.  ITS POSITION IS
0005  C  INDICATED IN THE LOWER LEFT HAND CORNER OF THE SCREEN.
0006  C
0007  C      IX = X-COORDINATE
0008  C      IY = Y-COORDINATE
0009  C
0010  C
0011  C      INTEGER WACO
0012  C
0013  C      DIMENSION ICR(7),IP00(5),IPXY(3)
0014  C
0015  C      EQUIVALENCE (ICR,ICR1),(ICR(2),ICR2),(ICR(3),ICR3),(ICR(5),I
0016  C      1 (ICR(6),ICR6),(ICR(7),ICR7),(IPXY,IPXY1),(IPXY(2),IPXY2)
0017  C
0018  C      DATA IP00/44000B,64000B,24015B,50017B,26002B/
0019  C      DATA ICR/0,0,0,22054B,0,0,0/
0020  C      DATA IPXY/0,0,24001B/
0021  C      DATA WACO,LEAO,LLAO/22000B,44000B,64000B/
0022  C
0023  C
0024  C
0025  C      WRITE POSITION ON SCREEN
0026  C
0027  C      CALL DRIVR(2,IP00,5)
0028  C
0029  C      ID1 = IY/100
0030  C      ICR1 = WACO + ID1 + 60B
0031  C      ID2 = (IY-ID1*100)/10
0032  C      ICR2 = WACO + ID2 + 60B
0033  C      ID3 = (IY-ID1*100-ID2*10)
0034  C      ICR3 = WACO + ID3 + 60B
0035  C      ID1 = IX/100
0036  C      ICR5 = WACO + ID1 + 60B
0037  C      ID2 = (IX-ID1*100)/10
0038  C      ICR6 = WACO + ID2 + 60B
0039  C      ID3 = IX-ID1*100-ID2*10
0040  C      ICR7 = WACO + ID3 + 60B
0041  C
0042  C      CALL DRIVR(2,ICR,7)
0043  C
0044  C      POSITION CURSOR
0045  C
0046  C      IPXY1 = IOR(LEAO,IAND(IX,777B))
0047  C      IPXY2 = IOR(LLAO,LAND(IY,377B))
0048  C      CALL DRIVR(2,IPXY,3)
0049  C      RETURN
0050  C      END
0051  $
```

&IMAGE T=00004 IS ON CRO0022 USING 00015 BLKS R=0161

```
0001  FTN4,Q,C,T
0002      PROGRAM IMAGE
0003  C
0004  C
0005  C  THIS PROGRAM IS THE IMAGE FILE MANAGER FOR THE IMAGE DISPLAY
0006  C  SUBSYSTEM.
0007  C
0008  C
0009      DIMENSION LU(5),IDCB1(272),IDCB2(528),IDCB3(144),JNAME(3),
0010      1 IFNAM(3),NAME(6),IDATA(512),KNAME(6)
0011  C
0012      INTEGER ENTRY(256),ISIZE(2),TEXT1(19)
0013  C
0014      EQUIVALENCE (ENTRY(7),NLINE),(ENTRY(8),NPIXL),(ENTRY(12),LOC
0015      1 (ENTRY(13),JNAME),(ENTRY(16),IFNAM),(ENTRY(19),IFNUM)
0016      2,(ENTRY,KNAME)
0017      2,(TEXT1,ENTRY(129))
0018      EQUIVALENCE (ISIZE(2),ISIZ2)
0019  C
0020  C
0021  C  GET INPUT PARAMETERS
0022  C
0023      CALL RMPAR(LU)
0024      IF (LU .LE. 0) LU = 1
0025  C
0026  C  OUTPUT HEADING
0027  C
0028  900  WRITE(LU,1)
0029  1      FORMAT(//      I M A G E   F I L E   M A N A G E R//)
0030  C
0031  C  GET COMMAND INPUT
0032  C
0033  1000  WRITE(LU,2)
0034  2      FORMAT(">_")
0035      READ(LU,3) ICMD
0036  3      FORMAT(A2)
0037  C
0038  C  EXECUTE COMMAND
0039  C
0040      IF (ICMD .NE. 2H??) GO TO 1010
0041  C
0042  C  COMMAND IS HELP
0043  C
0044      WRITE(LU,4)
0045  4      FORMAT(/" COMMANDS ARE: "/,
0046      1"  BU-BUILD IMAGE FILE"/,
0047      2"  DI-DISPLAY IMAGE ON GMR-27"/,
0048      3"  SA-SAVE IMAGE TO TAPE"/
0049      4"  RE-RESTORE IMAGE TO DISC"/,
0050      4"  DL-DIRECTORY LIST"/,
0051      4"  PU-PURGE IMAGE"/,
0052      4"  WT-WRITE NASA TAPE"/,
0053      5"  ??-HELP"/,
0054      6"  EX-EXIT"/)
0055      GO TO 1000
```

```
0056 C
0057 1010 IF (ICMD .NE. 2HBU) GO TO 1030
0058 C
0059 C BUILD IMAGE COMMAND
0060 C
0061     CALL EXEC(23+100000B,6HBLDIM ,LU)
0062     GO TO 1020
0063 S     GO TO 900
0064 C
0065 C PROGRAM NOT RP'ED
0066 C
0067 1020 WRITE(LU,6)
0068 6     FORMAT(" BLDIM NOT RP'ED!")
0069     GO TO 1000
0070 C
0071 1030 IF (ICMD .NE. 2HDI) GO TO 1045
0072 C
0073 C DISPLAY IMAGE COMMAND
0074 C
0075     CALL EXEC(23+100000B,6HDSPLY ,LU)
0076     GO TO 1040
0077 7     GO TO 900
0078 C
0079 C DSPLY NO RP'ED
0080 C
0081 1040 WRITE(LU,8)
0082 8     FORMAT(" DSPLY NOT RP'ED!")
0083     GO TO 1000
0084 C
0085 1045 IF (ICMD .NE. 2HWT) GO TO 1050
0086 C
0087 C WRITE NASA TAPE
0088 C
0089     CALL EXEC(23,6HWTAPE ,LU)
0090     GO TO 1000
0091 C
0092 C
0093 1050 IF (ICMD .NE. 2HSA .AND. ICMD .NE. 2HRE .AND.
0094 1     ICMD .NE. 2HPU) GO TO 1200
0095 C
0096 C SAVE/RESTORE IMAGE TO/FROM TAPE AND PURGE IMAGE
0097 C
0098 C OPEN DIRECTORY FILE
0099 C
0100     CALL OPEN(IDCBL,IERR,6HIMDIRC,2,2HIM,23,272)
0101     IF (IERR .LT. 0) GO TO 9999
0102 C
0103 C GET IMAGE NAME
0104 C
0105     WRITE(LU,9)
0106 9     FORMAT(" ENTER IMAGE NAME (12 CHARACTERS)? _")
0107     READ(LU,10) NAME
0108 10    FORMAT(6A2)
0109 C
0110 C FIND IMAGE
```

```
0111 C
0112 1060 CALL READF(IDCB1,IERR,ENTRY,256,LEN)
0113 IF (LEN .NE. -1) GO TO 1070
0114 C
0115 C EOF REACHED
0116 C
0117 WRITE(LU,11)
0118 11 FORMAT(" IMAGE NOT FOUND!")
0119 CALL CLOSE(IDCB1)
0120 GO TO 1000
0121 C
0122 1070 IF (IERR .LT. 0) GO TO 9999
0123 C
0124 C COMPARE NAME OF IMAGE
0125 C
0126 IF (ICMPW(ENTRY,NAME,6) .NE. C) GO TO 1060
0127 C
0128 C IMAGE FOUND
0129 C
0130 IF (ICMD .EQ. 2HRE) GO TO 1120
0131 IF (ICMD .EQ. 2HPU) GO TO 1300
0132 C
0133 C TASK IS TO SAVE IMAGE
0134 C
0135 IF (LOC .EQ. 1) GO TO 1090
0136 C
0137 C IMAGE ALREADY ON TAPE
0138 C
0139 WRITE(LU,12)
0140 12 FORMAT(" IMAGE NOT ON DISC!")
0141 GO TO 1000
0142 C
0143 C IMAGE ON DISC
0144 C
0145 1090 CALL OPEN(IDCB2,IERR,JNAME,0,0,0,528)
0146 IF (IERR .LT. 0) GO TO 9999
0147 C
0148 C GET TYPE O FILE
0149 C
0150 C WRITE(LU,13)
0151 C13 FORMAT(" TYPE MT LU 000# ?_")
0152 C READ(LU,14) IFNAM
0153 C14 FORMAT(3A2)
0154 IFNAM=2HLU
0155 IFNAM(2) =2H00
0156 IFNAM(3) =2H08
0157 WRITE(LU,131)
0158 131 FORMAT(" SELECT OPTION"/" 1. 8-BIT PACKED"/" 2. UNPACKED
0159 READ(LU,*) IPACK
0160 CALL OPEN(IDCB3,IERR,IFNAM)
0161 IF (IERR .LT. 0) GO TO 9999
0162 CALL RWNDF(IDCB3,IERR)
0163 IF (IERR .LT. 0) GO TO 9999
0164 WRITE(LU,15)
0165 15 FORMAT(" FILE #?_")
0166 READ(LU,*) IFNUM
0167 CALL SPACE(IDCB3,IERR,IFNUM-1)
0168 IF (IERR .LT. 0) GO TO 9999
```

```
0169 C
0170 C      WRITE HEADER ON TAPE
0171 C
0172      CALL WRITF(IDCB3,IERR,ENTRY,11)
0173      IF (IERR .LT. 0) GO TO 9999
0174 C
0175 C      NOW TRANSFER DATA
0176 C
0177      DO 1100 I = 1,NLINE
0178      CALL READF(IDCB2,IERR,ILATA,512)
0179      IF (IERR .LT. 0) GO TO 9999
0180 C
0181      IF(IPACK .NE. 1) GO TO 1101
0182 C
0183 C      PACK DATA
0184 C
0185      DO 1102 J =1,NPIXL,2
0186      K = 0.5*(J+1)
0187      IVAR=IDATA(J+1)
0188      CALL ROT8(IVAR,KVAR)
0189 1102  IDATA(K)=IOR(IDATA(J),KVAR)
0190 1101  CALL WRITF(IDCB3,IERR, IDATA,NPIXL)
0191      IF (IERR .LT. 0) GO TO 9999
0192 1100  CONTINUE
0193 C
0194 C      WRITE EOF
0195 C
0196      CALL WRITF(IDCB3,IERR,0,-1)
0197      IF (IERR .LT. 0) GO TO 9999
0198 C
0199 C      PURGE DISC FILE
0200 C
0201      CALL PURGE(IDCB2,IERR,JNAME,2HIM)
0202      IF (IERR .LT. 0) GO TO 9999
0203 C
0204 C      UPDATE ENTRY
0205 C
0206      LOC = 2
0207 C
0208      CALL POSNT(IDCB1,IERR,-1)
0209      IF (IERR .LT. 0) GO TO 9999
0210      CALL WRITF(IDCB1,IERR,ENTRY,256)
0211      IF (IERR .LT. 0) GO TO 9999
0212 C
0213      CALL CLOSE(IDCB1)
0214      CALL RWNDF(IDCB3)
0215      CALL CLOSE(IDCB3)
0216      GO TO 1000
0217 C
0218 C      RESTORE IMAGE FROM TAPE
0219 C
0220 1120  IF (LOC .EQ. 2) GO TO 1130
0221 C
0222 C      IMAGE ON DISC
0223 C
0224      WRITE(LU,16)
0225 16      FORMAT(" IMAGE ALREADY ON DISC!")
0226      CALL CLOSE(IDCB1)
0227      GO TO 1000
```

```
0228 C
0229 C  CREATE DISC FILE
0230 C
0231 1130 ISIZE = (FLOAT(NLINE)*FLOAT(NPIXL)+127.)/128.
0232 ISIZ2 = NPIXL
0233 C
0234     CALL CREAT(IDCB2,IERR,JNAME,ISIZE,2,2HIM,23,528)
0235     IF (IERR .GE. 0) GO TO 1135
0236 C
0237 C  CAN'T CREATE DISC FILE
0238 C
0239     WRITE(LU,19)
0240 19     FORMAT(" CAN'T CREATE DISC FILE!!")
0241     CALL CLOSE(IDCB1)
0242     GO TO 1000
0243 C
0244 C  OPEN TYPE 0 FILE
0245 C
0246 1135 CALL OPEN(IDCB3,IERR,IFNAM)
0247 C
0248 C  GET LU OF TYPE 0 FILE
0249 C
0250     CALL LOCF(IDCB3,IERR,IREC,IRB,IOFF,JSEC,MTLU)
0251     IF (IERR .LT. 0) GO TO 9999
0252 C
0253 C  TELL USER TO MOUNT TAPE
0254 C
0255     WRITE(LU,17) MTLU
0256 17     FORMAT(" MOUNT TAPE ON LU ",I2" ENTER RETURN WHEN READY")
0257     CALL EXEC(1,LU,IREC,1)
0258 C
0259 C  REWIND TAPE
0260 C
0261     CALL RWNDF(IDCB3,IERR)
0262     IF (IERR .LT. 0) GO TO 9999
0263 C
0264 C  SPACE FORWARD TO FILE
0265 C
0266     CALL SPACE(IDCB3,IERR,IFNUM-1)
0267 C
0268 C  READ HEADER
0269 C
0270     CALL READF(IDCB3,IERR,IData,11)
0271     IF (IERR .LT. 0) GO TO 9999
0272     IF(ICMPW(IDATA,ENTRY,11) .NE. 0) GO TO 1160
0273 C
0274 C  HEADER COMPARES
0275 C
0276 C  TRANSFER DATA
0277 C
0278     DO 1140 I=1,NLINE
0279     CALL READF(IDCB3,IERR,IData,NPIXL)
0280     IF (IERR .LT. 0) GO TO 9999
0281     CALL WRITF(IDCB2,IERR,IData,NPIXL)
0282     IF (IERR .LT. 0) GO TO 9999
0283 1140 CONTINUE
0284 C
```

```
0285      CALL RWNDF(IDCB3)
0286      CALL CLOSE(IDCB3)
0287      CALL CLOSE(IDCB2)
0288 C
0289 C  UPDATE DIRECTORY ENTRY
0290 C
0291      LOC = 1
0292 C
0293      CALL POSNT(IDCB1,IERR,-1)
0294      IF (IERR .LT. 0) GO TO 9999
0295      CALL WRITF(IDCB1,IERR,ENTRY,256)
0296      IF (IERR .LT. 0) GO TO 9999
0297      CALL CLOSE(IDCB1,IERR)
0298      IF (IERR .LT. 0) GO TO 9999
0299 C
0300      GO TO 1000
0301 C
0302 C  LABEL DOES NO MATCH
0303 C
0304 1160  WRITE(LU,18)
0305 18   FORMAT(" WRONG FILE!!")
0306      CALL RWNDF(IDCB3)
0307      CALL CLOSE(IDCB3)
0308      CALL CLOSE(IDCB1)
0309      GO TO 1000
0310 C
0311 C
0312 1200  IF (ICMD .NE. 2HDL) GO TO 1230
0313 C
0314 C  DIRECTORY LIST
0315 C
0316 C  OPEN DIRECTORY FILE
0317 C
0318      CALL OPEN(IDCB1,IERR,6H1MDIRC)
0319      IF (IERR .LT. 0) GO TO 9999
0320 C
0321 C  OUTPUT HEADING
0322 C
0323      WRITE(LU,30)
0324 30   FORMAT(//IMAGE NAME      #LINES  #PIXELS  LOC      TEXT")
0325 C
0326 C  OUTPUT INFO
0327 C
0328 1210  CALL READF(IDCB1,IERR,ENTRY,256,LEN)
0329      IF (LEN .NE. -1) GO TO 1220
0330 C
0331 C  EOF REACHED
0332 C
0333      CALL CLOSE(IDCB1)
0334      GO TO 1000
0335 C
0336 1220  IF (IERR .LT. 0) GO TO 9999
0337 C
0338      IF (ENTRY .EQ. -1) GO TO 1210
0339      ICHR = 2HD
0340      IF (LOC .NE. 1) ICHR = 2HT
0341      WRITE(LU,31)KNAME,NLINE,NPIXL,ICHR,TEXT1
0342 31   FORMAT(6A2,2X,I4,4X,I4,3X,A5,2X,!9A2)
0343      GO TO 1210
```

```
0344 C
0345 1230 IF (ICMD .EQ. 2HEX) GO TO 1240
0346 C
0347 C ILLEGAL COMMAND
0348 C
0349      WRITE(LU,22)
0350 22      FORMAT("ILLEGAL COMMAND!")
0351      GO TO 1000
0352 C
0353 1240 WRITE(LU,23)
0354 23      FORMAT("END PROGRAM")
0355      CALL EXEC(6)
0356 C
0357 C PURGE FILE
0358 C
0359 1300 CALL POSNT(IDCBL,IERR,-1)
0360      IF (IERR .LT. 0) GO TO 9999
0361      ENTRY = -1
0362      CALL WRITF(IDCBL,IERR,ENTRY,256)
0363      IF (IERR .LT. 0) GO TO 9999
0364 C
0365 C PURGE DATA FILE
0366 C
0367      CALL PURGE(IDCBL,IERR,JNAME,2HIM)
0368      CALL CLOSE(IDCBL)
0369      GO TO 1000
0370 C
0371 C
0372 C ERROR
0373 C
0374 C
0375 9999 WRITE(LU,20) IERR
0376 20      FORMAT(" FILE ERROR ",I6)
0377      CALL CLOSE(IDCBL)
0378      GO TO 1000
0379      END
0380 $
```

&SPACE T=00004 IS ON CRO0022 USING 00004 BLKS R=0029

```
0001  FTN4
0002      SUBROUTINE SPACE(IDCB,IERR,NUM)
0003  C
0004  C  THIS SUBROUTINE IS USED TO SPACE FORWARD OR BACKWARD THE
0005  C  NUMBER OF FILES SPECIFIED.
0006  C
0007      DIMENSION IDCB(144)
0008  C
0009      DATA IFRWD,IBACK/1300B,1400B/
0010  C
0011  C
0012  C
0013      IERR = 0
0014      IF (NUM .EQ. 0) RETURN
0015  C
0016      IDIR = IFRWD
0017      IF (NUM .GT. 0) GO TO 100
0018      IDIR = IBACK
0019      NUM = -NUM
0020  C
0021  C
0022  100  DO 110 I=1,NUM
0023      CALL FCONT(IDCB,IERR,DIR)
0024      IF (IERR .LT. 0) RETURN
0025  110  CONTINUE
0026  C
0027      RETURN
0028      END
0029  $
```

&RESIZ T=00004 IS ON CROC022 USING 00060 BLKS R=0541

```
0001  FTN4,Q,T,C
0002      PROGRAM RESIZ
0003  C      WRITTEN BY W. E. ALEXANDER
0004  C
0005  C      PROGRAM FORMS A PART OF THE SPATIAL DOMAIN FILTERING PACKAGE
0006  C
0007  C      PROGRAM ALLOWS THE USER TO INTERPOLATE AND SCALE AN IMAGE AND
0008  C      CHANGE ITS DATA TYPE.  THUS A FLOATING POINT IMAGE CAN BE MADE
0009  C      INTO AN EIGHT BIT IMAGE.
0010  C
0011  C
0012  C
0013      DIMENSION F(512),G(512),IOP(512),IPRAM(5),NSON(3,2)
0014      DIMENSION A(3,2,2),B(3,2,2),NAME(3),INM(3)
0015      DIMENSION JMES(40),DIRC(515),IKTM(5)
0016      INTEGER WFINT,READL,RITEL
0017      EQUIVALENCE(G(1),IOP(1))
0018      EQUIVALENCE(F(1),G(1))
0019      EQUIVALENCE(DIRC(4),F(1)),(DIRC(1),INM(1)),(INM(1),NROW)
0020      EQUIVALENCE(INM(2),ICOLS),(DIRC(2),AMAX),(DIRC(3),AMIN)
0021  C
0022      DATA NSON/2HLF,2HLT,2HR ,2HDI,2HNT,2HP /
0023      CALL RMPAR(IPRAM)
0024      LU=IPRAM(1)
0025      IF(LU.EQ.0) LU=1
0026  C
0027  C      INITIALIZE PARAMETERS
0028  C
0029      ITYPE = 8
0030  C
0031      CALL CODE
0032      WRITE (JMES,6)
0033  6      FORMAT (" RESIZE")
0034      CALL TRMGN (JMES,LU,0)
0035      CALL BLANK (JMES)
0036      NTYPE=32
0037      IRTCD=0
0038      IMXX=512
0039      IMXP1=IMXX+1
0040      IFE=0
0041      ILE=511
0042      IFR=0
0043      ILR=511
0044  C
0045  5      CALL CODE
0046      WRITE (JMES,995)
0047  995      FORMAT (" RESIZE IMAGE ")
0048      CALL TRMGN (JMES,LU,0)
0049      CALL BLANK (JMES)
0050  C
0051  C      SPECIFY DATA LENGTH FOR OUTPUT
0052  C
```

```
0053    10 CALL CODE
0054    WRITE(JMES,11)
0055    11 FORMAT(" SPECIFY OUTPUT DATA TYPE")
0056    CALL TRMGN(JMES,LU,0)
0057    CALL BLANK(JMES)
0058    CALL CODE
0059    WRITE(JMES,12)
0060    12 FORMAT(" 1. 8 BIT IMAGE")
0061    CALL TRMGN(JMES,LU,0)
0062    CALL BLANK(JMES)
0063    CALL CODE
0064    WRITE(JMES,13)
0065    13 FORMAT(" 2. 15 BIT IMAGE")
0066    CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0067    C
0068    IRTM=ICD
0069    C
0070    CALL BLANK(JMES)
0071    15 CALL SPCHR (IRTM,IRT)
0072    GO TO (500,10,5,20,17,17),IRT
0073    17 CALL CKFLD(2,ICD,IRS)
0074    GO TO (25,25,30,30,20),IRS
0075    20 IW=1
0076    GO TO 950
0077    25 ITYPE =8
0078    IMAX=255
0079    GO TO 32
0080    30 ITYPE =15
0081    IMAX=32767
0082    C
0083    C      SPECIFY WORK FILE
0084    C
0085    C
0086    32 CALL BLANK(JMES)
0087    CALL CODE
0088    WRITE(JMES,450)
0089    450 FORMAT(" SELECT OPTION")
0090    CALL TRMGN(JMES,LU,0)
0091    CALL BLANK(JMES)
0092    CALL CODE
0093    WRITE(JMES,455)
0094    455 FORMAT(" 1. SPECIFY NEW IMAGE")
0095    CALL TRMGN(JMES,LU,0)
0096    CALL BLANK(JMES)
0097    CALL CODE
0098    WRITE(JMES,460)
0099    460 FORMAT(" 2. USE CURRENT WORK FILE")
0100    CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0101    CALL BLANK(JMES)
0102    465 CALL CKFLD(2,ICD,IRS)
0103    GO TO (475,475,480,485),IRS
0104    485 IW=12
0105    C
0106    C      OPEN WORK FILE
0107    C
```

```

      .. 40 TO 900
0109  475 IGET=WFINT (NROW,ICOLS,AMAX,AMIN,LU)
0110    IF(IGET.LT.0) GO TO 999
0111  480 IGET=READL(-1,0,511,DIRC)
0112    IF(IGET.LT.0) GO TO 999
0113  40 CALL CODE
0114    WRITE(JMES,45) AMAX,AMIN
0115  45 FORMAT(" AMAX= ",E12.5,5X," AMIN= ",E12.5,5X,"FOR IMAGE")
0116    CALL TRMGN(JMES,LU,0)
0117    CALL BLANK(JMES)
0118 C
0119 C      SPECIFY IMAGE SCALING OPTION
0120 C
0121  50 CALL CODE
0122    WRITE(JMES,51)
0123  51 FORMAT("      SPECIFY IMAGE SCALING OPTION")
0124    CALL TRMGN(JMES,LU,0)
0125    CALL BLANK(JMES)
0126    CALL CODE
0127    WRITE(JMES,52)
0128  52 FORMAT(" 1. AUTOMATIC SCALING")
0129    CALL TRMGN(JMES,LU,0)
0130    CALL BLANK(JMES)
0131    CALL CODE
0132    WRITE(JMES,53)
0133  53 FORMAT(" 2. SYSTEM DEFAULT OPTION")
0134    CALL TRMGN(JMES,LU,0)
0135    CALL BLANK(JMES)
0136    CALL CODE
0137    WRITE(JMES,54)
0138  54 FORMAT(" 3. USER SPECIFIED SCALE FACTOR")
0139    CALL TRMGN(JMES,LU,0)
0140    CALL BLANK(JMES)
0141    CALL CODE
0142    WRITE(JMES,56)
0143  56 FORMAT(" 4. USER SPECIFIED MAX AND MIN")
0144    CALL TRMGN(JMES,LU,0)
0145    CALL BLANK(JMES)
0146    CALL CODE
0147    WRITE(JMES,57)
0148  57 FORMAT(" 5. LOG COMPRESSION")
0149    CALL TRMGN(JMES,LU,0)
0150    CALL BLANK(JMES)
0151    CALL CODE
0152    WRITE(JMES,58)
0153  58 FORMAT(" 6. EXPONENTIATION OPTION")
0154    CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0155    CALL BLANK(JMES)
0156  55 CALL CKFLD (6,ICD,IRS)
0157    GO TO (65,65,75,80,105,160,165,60),IRS
0158  60 IW=2
0159    GO TO 950
0160 C
0161 C      AUTOMATIC SCALING SELECTED
0162 C
0163  65 SCL=AMAX-AMIN
0164    IF (ABS(SCL).LE.1.0E-5) GO TO 70
0165    SCL=FLOAT(IMAX)/SCL
0166    IOPT=1
0167    GO TO 190
0168 C
0169 C      SYSTEM DEFAULT SCALING OPTION SELECTED
0170 C
0171  75 SCL=1.0
0172    IOPT=2
0173    GO TO 100

```

ORIGINAL PAGE IS
OF POOR QUALITY

```

0234      CALL CODE
0235      WRITE(JMES,151)
0236 151 FORMAT(" SCALE TOO SMALL")
0237      CALL TRMGN(JMES,LU,0)
0238      CALL BLANK(JMES)
0239      CALL CODE
0240      WRITE(JMES,152)
0241 152 FORMAT(" ENTER CR TO GO TO SYSTEM LEVEL MENU")
0242      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0243      CALL BLANK(JMES)
0244      IRTCD=1HX
0245      GO TO 1000
0246 155 SCL=(1.0/SCL)*FLOAT(IMAX)
0247      GO TO 190
0248 C
0249 C      LOG COMPRESSION OPTION SELECTED
0250 C
0251 160 CALL CODE
0252      WRITE(JMES,161)
0253 161 FORMAT(" LOG COMPRESSION OPTION SELECTED")
0254      CALL TRMGN(JMES,LU,0)
0255      CALL BLANK(JMES)
0256      IOPT=5
0257      SCL=FLOAT(IMAX)/ALOG(AMAX-AMIN+1.0)
0258      GO TO 190
0259 C
0260 C      EXPONENTIATION OPTION SELECTED
0261 C
0262 165 CALL CODE
0263      WRITE(JMES,166)
0264 166 FORMAT(" ENTER DESIRED EXPONENT")
0265      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0266      CALL BLANK(JMES)
0267 170 CALL SPCHR (IRTM,IRT)
0268      POWER=RTM
0269      IF (IRT .EQ. 5) GO TO 180
0270 175 LW=6
0271      GO TO 950
0272 180 POWER=ABS(POWER)
0273      CALL CODE
0274      WRITE(JMES,185) POWER
0275 185 FORMAT(" EXPONENT= ",1F10.4)
0276      CALL TRMGN(JMES,LU,0)
0277      CALL BLANK(JMES)
0278      SCL=FLOAT(IMAX)/((AMAX-AMIN)**POWER)
0279      IOPT=6
0280 C
0281 C      OBTAIN PARAMETERS FOR RESIZING IMAGE
0282 C
0283 190 INUM=64
0284      INCNT=0
0285      IMCNT=0
0286      NTST=INUM
0287 195 CALL CODE
0288      WRITE(JMES,196)
0289 196 FORMAT(" INDEPENDENT DIRECTIONAL SCALING")
0290      CALL TRMGN(JMES,LU,0)
0291      CALL BLANK(JMES)

```

```

0174 C
0175 C      USER ENTERS SCALE FACTOR
0176 C
0177 80  CALL CODE
0178      WRITE(JMES,81)
0179 81  FORMAT(" ENTER DESIRED SCALE FACTOR")
0180      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0181      CALL BLANK(JMES)
0182 85  CALL SPCHR (IRTM,IRT)
0183      GO TO (1000,1000,1000,1000,855),IRT
0184 855  GAIN=ABS(RTM)
0185  IF (IRT .EQ. 5) GO TO 95
0186 90  IW=3
0187  GO TO 950
0188 95  CALL CODE
0189      WRITE(JMES,100) GAIN
0190 100  FORMAT ("  SCALE FACTOR = ",F10.4)
0191      CALL TRMGN(JMES,LU,0)
0192      CALL BLANK(JMES)
0193      IOPT=3
0194      SCL=GAIN
0195  GO TO 190
0196 C
0197 C      USER SPECIFIED MAXIMUM AND MINIMUM
0198 C
0199 105  IOPT=4
0200      CALL CODE
0201      WRITE(JMES,106)
0202 106  FORMAT("  ENTER MAXIMUM FOR IMAGE")
0203      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0204      CALL BLANK(JMES)
0205 110  CALL SPCHR (IRTM,IRT)
0206      GO TO (1000,1000,1000,1000,910),IRT
0207 910  AMXIN=RTM
0208  IF(RTM.NE.0B) GO TO 120
0209 115  IW=4
0210  GO TO 950
0211 120  CALL CODE
0212      WRITE(JMES,121) AMXIN
0213 121  FORMAT("  MAXIMUM FOR IMAGE = ",1PE15.8)
0214      CALL TRMGN(JMES,LU,0)
0215      CALL BLANK(JMES)
0216 130  CALL CODE
0217      WRITE(JMES,131)
0218 131  FORMAT("  ENTER MINIMUM FOR IMAGE")
0219      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0220      CALL BLANK(JMES)
0221 135  CALL SPCHR (IRTM,IRT)
0222      GO TO (1000,1000,1000,1000,935),IRT
0223 935  AMNIN=RTM
0224  IF (IRT .EQ. 5) GO TO 145
0225 140  IW=5
0226  GO TO 950
0227 145  CALL CODE
0228      WRITE(JMES,150) AMNIN
0229 150  FORMAT ("-- MINIMUM FOR IMAGE =",F10.4)
0230      CALL TRMGN(JMES,LU,0)
0231      CALL BLANK(JMES)
0232      SCL=AMXIN-AMNIN
0233  IF(SCL.GE.1.0E-5) GO TO 155

```

```

0292 931 CALL CODE
0293   WRITE(JMES,197)
0294   197 FORMAT(" ENTER ROW SCALE FACTOR")
0295   CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0296   CALL BLANK(JMES)
0297   200 CALL SPCHR (IRTM,IRT)
0298   YS=RTM
0299   IF (IRT .EQ. 5) GO TO 210
0300   205 IW=7
0301   GO TO 950
0302   210 CALL CODE
0303   WRITE(JMES,211)
0304   211 FORMAT(" ENTER COLUMN SCALE FACTOR ")
0305   CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0306   CALL BLANK(JMES)
0307   215 CALL SPCHR (IRTM,IWT)
0308   XS=RTM
0309   IF (IRT .EQ. 5) GO TO 225
0310   220 IW=8
0311   GO TO 950
0312 C
0313 C      CHECK TO SEE IF INTERPOLATION IS REQUIRED.  IF NOT BRANCH.
0314 C
0315 C
0316 C      COMPUTE NEW SIZE OF IMAGE
0317 C
0318   225 NNEW=YS*NROW+.5
0319   NCOLS=XS*ICOLS+.5
0320 C
0321   IF(XS.EQ.1.0.AND.YS.EQ.1.0) GO TO 260
0322   IF(NCOLS.LE.512) GO TO 230
0323   CALL CODE
0324   WRITE(JMES,690) NCOLS,XS
0325   690 FORMAT(" CALCULATED COLUMN SIZE = ",1I5,"(SF =",1F5.2")")
0326   CALL TRMGN(JMES,LU,0)
0327   CALL BLANK(JMES)
0328   CALL CODE
0329   WRITE (JMES,969)
0330   969 FORMAT ("      512 IS MAXIMUM ALLOWABLE OUTPUT")
0331   CALL TRMGN (JMES,LJ,0)
0332   CALL BLANK(JMES)
0333   CALL CODE
0334   WRITE(JMES,226)
0335   226 FORMAT(" REENTER COLUMN SCALE FACTOR")
0336   CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0337   CALL BLANK(JMES)
0338   GO TO 215
0339   230 CALL CODE
0340   WRITE(JMES,695) NNEW,YS,NCOLS,XS
0341   695 FORMAT("-- OUTPUT IMAGE-",14X,"ROWS = ",1I5,"(SF=",1F5.2,",
0342   *14X,"COLUMNS = ",1I5," (SF = ",1F5.2,")")
0343   231 CALL TRMGN(JMES,LU,0)
0344   CALL BLANK(JMES)
0345   CALL CODE
0346   WRITE(JMES,1232)
0347   1232 FORMAT(" 1. VALUES OKAY")
0348   CALL TRMGN(JMES,LU,0)
0349   CALL BLANK(JMES)

```

```

0350      CALL CODE
0351      WRITE(JMES,1233)
0352 1233 FORMAT(" 2. REENTER SCALE FACTOR")
0353      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0354      CALL BLANK(JMES)
0355 232   CALL SPCHR (IRTM,IRT)
0356      CALL CKFLD(2,ICD,IRS)
0357      GO TO (234,234,931),IRS
0358 233   IW=9
0359      GO TO 950
0360 C
0361 C      COMPUTE INCREMENTS FOR INTERPOLATION
0362 C
0363 234   MM=NNEW
0364      IFLT=0
0365      DY=FLOAT(NROW)/FLOAT(NNEW)
0366      DX=FLOAT(ICOLS)/FLOAT(NCOLS)
0367      IF(DY.LE.1.0) GO TO 235
0368 700   CALL CODE
0369      WRITE(JMES,1750)
0370 1750 FORMAT(" IMAGE SHOULD BE FILTERED BEFORE INTERPOLATION")
0371      CALL TRMGN(JMES,LU,0)
0372      CALL BLANK(JMES)
0373      CALL CODE
0374      WRITE (JMES,9750)
0375 9750 FORMAT (" TO PREVENT ALIASING")
0376      CALL TRMGN(JMES,LU,0)
0377      CALL BLANK(JMES)
0378      CALL CODE
0379      WRITE(JMES,760)
0380 760   FORMAT(" 1. CONTINUE ")
0381      CALL TRMGN(JMES,LU,0)
0382      CALL BLANK(JMES)
0383      WRITE(JMES,761)
0384 761   FORMAT(" 2. PREFILTER IMAGE")
0385      CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0386 704   CALL SPCHR (IRTM,IRT)
0387      GO TO (1000,1000,1000,1000,904),IRT
0388 904   CALL CKFLD(2,ICD,IRS)
0389      GO TO (702,702,710),IRS
0390 710   IW=10
0391      GO TO 950
0392 C
0393 C      SCHELULE FILTER TO PREVENT ALIASING
0394 C
0395 702   FCX=0.8*FLOAT(NCOLS)/FLOAT(ICOLS)
0396      FCY=0.8*FLOAT(NNEW)/FLOAT(NROW)
0397      NX=2
0398      NY=2
0399      ITME=0
0400      CALL XYFLT(U,V,FCX,FCY,NX,NY,N,A,B)
0401 C
0402      DO 705 II=1,3
0403 705  NAME(II)=NSON(II,1)
0404      CALL EXEC (9,NAME,LU,0,NCOLS-1)
0405 C
0406 C      INITIIZE FOR RESIZING

```

```

0407 C
0408 135 CALL CODE
0409      WRITE(JMES,810)
0410 810 FORMAT(" RESIZING OF IMAGE IN PROGRESS")
0411      CALL TRMGN(JMES,LU,0)
0412      CALL BLANK(JMES)
0413 C
0414 C      SCHEDULE DINTP FOR RESIZING IMAGE
0415 C
0416      INCW=LU+200B
0417      DO 250 II=1,3
0418 250 NAME(II)=NSON(II,2)
0419 C
0420 C      CLOSE WORK FILE
0421 C
0422      CALL CLSWF(NROW,ICOLS,AMAX,AMIN)
0423      CALL EXEC(13,ICNW,IPRAM(3),IPRAM(4),IPRAM(5))
0424      CALL EXEC(23,NAME,IPRAM(1),NNEW,NCOLS,IPRAM(4),IPRAM(5))
0425 C
0426 C      OPEN WORK FILE
0427 C
0428 C
0429 C
0430 C      CALL OPEN(IDCB,IGET,6HWF0000,2,0,0,528)
0431 C      IF(IGET.LT.0) GO TO 999
0432 C
0433 C      REMAP INTENSITY VALUES FOR IMAGE
0434 C
0435 C      OBTAIN NEW SIZE PARAMETERS
0436 C
0437 C
0438      IGET=READL(-1,0,511,DIRC)
0439      IF(IGET.LT.0) GO TO 999
0440 C
0441 260 IMCNT=0
0442 IZCNT=0
0443 DO 405 NN=1,NROW
0444 C
0445 C      READ IN NEW ROW
0446 C
0447      NNM1=NN-1
0448      IGET=READL(NNM1,IFE,ICOLS-1,F)
0449      IF(IGET.LT.0) GO TO 999
0450      IF (IOPT .GT. 6) GO TO 310
0451      GO TO (306,310,320,330,340,350),IOPT
0452 306 DO 305 I=1,ICOLS
0453 305 G(I)=(F(I)-AMIN)*SCL+0.5
0454      GO TO 360
0455 310 DO 315 I=1,ICOLS
0456 315 G(I)=F(I)+0.5
0457      GO TO 360
0458 320 DO 325 I=1,ICOLS
0459 325 G(I)=(F(I)*SCL+0.5)
0460      GO TO 360
0461 330 DO 335 I=1,ICOLS
0462 335 G(I)=(F(I)-AMIN)*SCL+0.5
0463      GO TO 360
0464 340 DO 345 I=1,ICOLS
0465      F(I)=AMAX1(F(I),AMIN)
0466 345 G(I)=SCL*(ALOG(F(I)-AMIN+1.0))+0.5
0467      GO TO 360
0468 350 DO 355 I=1,ICOLS
0469 355 G(I)=SCL*(F(I)-AMIN)**POWER+0.5

```

```

0470 C
0471 C      WRITE OUTPUT TO WORK FILE
0472 C
0473 360 IGET=RITEL(NNM1,0,ICOLS-1,G)
0474 IF(IGET.LT.0) GO TO 999
0475 C
0476 C      IF OUTPUT IS 8 BIT, WRITE TO DISPLAY
0477 C
0478 IF(ITYPE.EQ.15) GO TO 365
0479 C
0480 DO 370 I=1,ICOLS
0481 IF(G(I).GT.(FLOAT(IMAX)+0.5))IMCNT=IMCNT+1
0482 IF(G(I).LT.0.0) IZCNT=IZCNT+1
0483 IOP(I)=MIN0(IFIX(G(I)),IMAX)
0484 370 IOP(I)=MAX0(IOP(I),0)
0485 365 IF(NN.LT.NTST) GO TO 400
0486 NTST=NTST+INUM
0487 CALL CODE
0488 WRITE(JMES,375) NN,NNEW
0489 375 FORMAT("- RESIZE ROWS DONE/ TO DO ",1I4,"/",1I4,/)
0490 CALL TRMGN(JMES,LU,0)
0491 CALL BLANK(JMES)
0492 C
0493 400 IF(ITYPE.NE.8) GO TO 405
0494 IGET=WLINE(NNM1,00,ICOLS-1,IOP)
0495 IF(IGET.LT.0) GO TO 999
0496 405 CONTINUE
0497 C
0498 C      CLOSE WORK FILE
0499 C
0500 AMAX=FLOAT(IMAX)
0501 AMIN=0.0
0502 CALL CLSWF(NROW,ICOLS,AMAX,AMIN)
0503 IRTCD=0
0504 ITOT=NROW*ICOLS
0505 ATOT=100.0/FLOAT(ITOT)
0506 PZERO=ATOT*FLOAT(IZCNT)
0507 PMAX=ATOT*FLOAT(IMCNT)
0508 IF(PZERO.EQ.0.0 .AND.PMAX.EQ.0.0) GO TO 1000
0509 CALL CODE
0510 WRITE(JMES,410) PZERO,PMAX
0511 CALL TRMGN(JMES,LU,0)
0512 CALL BLANK(JMES)
0513 410 FORMAT("- PERCENT CLIPPED AT ZERO =",F6.2,
0514 1" - PERCENT CLIPPED AT MAX =",F6.2)
0515 420 CALL TRMGN(JMES,LU,0)
0516 CALL BLANK(JMES)
0517 CALL CODE
0518 WRITE(JMES,421)
0519 421 FORMAT(" 1. CONTINUE[ 2. RESCALE IMAGE")
0520 CALL TRMGN(JMES,LU,1,RTM,ICD,IRTM)
0521 CALL BLANK(JMES)
0522 425 CALL SPCHR (IRTM,IRT)
0523 GO TO (1000,1000,1000,1000,925),IRT
0524 925 CALL CKFLD(2,ICD,IRS)
0525 GO TO (1000,1000,440,430),IRS
0526 430 IW=11
0527 GO TO 950

```

```
0528 440 CALL READL(-1,0,511,DIRC)
0529      GO TO 5
0530 70  CALL CODE
0531      WRITE(JMES,921)
0532 921  FORMAT (' SCALING SIZE ERROR')
0533      CALL TRMGN(JMES,LU,0)
0534      CALL BLANK(JMES)
0535 500  CONTINUE
0536      GO TO 999
0537 950  CALL CODE
0538      WRITE (JMES,21)
0539 21   FORMAT (" /INVALID SELECTION/")
0540      CALL TRMGN (JMES,LU,1,RTM,ICD,IRTM)
0541      CALL BLANK (JMES)
0542      GO TO (15,55,85,110,135,170,200,215,232,704,425),IW
0543 999  IF(IGET.EQ.-8) CALL CLSWF(NROW,ICOLS,AMAX,AMIN)
0544 1000 CALL EXEC(6)
0545      END
0546  $
0547  $
```

&ROT8 T=00004 IS ON CRO0022 USING 00002 BLKS R=0014

```
0001 ASMB,R,L,C
0002      NAM ROT8,6
0003      ENT ROT8
0004      EXT .ENTR
0005 *
0006 WORD BSS 1
0007 OUT  BSS 1
0008 *
0009 ROT8 NOP
0010      JSB .ENTR
0011      DEF WORD
0012      LDA WORD,I
0013      ALF,ALF
0014      STA OUT,I
0015      JMP ROT8,I
0016      END
```

&TRMGN T=00004 IS ON CRO0022 USING 00056 BLKS R=0439

```

0001  FTN4,L
0002      SUBROUTINE TRMGN(JMES,LU,IP,RTM,ICD,IRTM)
0003  C
0004  C  THIS SUBROUTINE IS USED TO WRITE OUT AND POSSIBLY READ BACK
0005  C  FROM THE TERMINAL INFORMATION NECESSARY FOR PROGRAM CONTROL.
0006  C  JMES IS THE MESSAGE TO BE OUTPUT TO THE LU. IP (IF =0) MEANS
0007  C  WRITE ONLY, (IF =1) MEANS TO WAIT FOR A RESPONSE FROM THE OPERATOR
0008  C  RTM IS THE RETURN FOR REAL NUMBERS, ICD IS A RETURN FOR INTEGER
0009  C  IRTM IS THE RETURN FOR ASCII CHARACTERS. ALL THREE TYPES OF RESULTS
0010  C  ARE GENERATED EACH TIME THIS SUBROUTINE IS CALLED. THE MAXIMUM
0011  C  OUTPUT MESSAGE IS 80 CHARACTERS LONG. THE MAXIMUM INPUT MESSAGE
0012  C  IS 10 CHARACTERS LONG.
0013  C
0014      DIMENSION JMES(40),IRTM(5)
0015      ICNWD=400B+LU
0016  C  WRITE THE MESSAGE TO THE LU
0017      CALL EXEC (2,ICNWD,JMES,40)
0018      IF (IP .EQ. 0) RETURN
0019  C  READ THE MESSAGE BACK FROM THE LU
0020      CALL EXEC (1,ICNWD,IRTM,5)
0021      CALL CODE
0022      READ (IRTM,*)ICD
0023      CALL CODE
0024      READ (IRTM,*)RTM
0025      RETURN
0026      END
0027      SUBROUTINE XYFLT(U,V,FCX,FCY,NX,NY,N,A,B)
0028  C
0029  C  WRITTEN BY W.E.ALEXANDER
0030  C  SUBROUTINE FORM A PART OF THE SPATIAL
0031  C  DOMAIN FILTERING PACKAGE.
0032  C  LOW PASS RECURSIVE FILTER DESIGN ROTINE
0033  C  WITH FCX NOT EQUAL TO FCY.
0034  C  FCX=RCX*S/PI WHERE RCX IS THE CUTOFF
0035  C  FREQUENCY IN THE X DIRECTION AND S IS THE SAMPLING
0036  C  INTERVAL (X DIRECTION)
0037  C  FCY = RCY*S/PI WHERE RCY IS THE CUTOFF
0038  C  THE Y DIRECTION AND S IS THE SAMPLING INTERVAL
0039  C  (Y DIRECTION)
0040  C  (0.010.LE.FCX.LE.0.950)
0041  C  (0.010.LE.FCY.LE.0.950)
0042  C
0043      COMPLEX P
0044      DIMENSION U(3,3,2),V(3,3,2),A(3,2,2),B(3,2,2)
0045  C
0046      PI=3.141592654
0047      D = 1.0E-10
0048      N = 3
0049      IF(NX.LE.2.AND.NY.LE.2)N=2
0050      EPS = 1.0
0051      DO 6 I=1,18
0052      IF (I.GT. 12) GO TO 7
0053      A(I)=0
0054      B(I)=0
0055      7      U(I)=0
0056      6      V(I)=0
0057  C

```

```

0058      A(1,2,1) = 1.0
0059      A(1,2,2) = 1.0
0060      B(1,2,1) = 1.0
0061      B(1,2,2) = 1.0
0062      C
0063      NXP = NX-1
0064      TX = SIN(PI*FCX*0.5)/COS(PI*FCX*0.5)
0065      TX = TX**2
0066      IF(TX.LE.D)TX=D
0067      CNX = TX**NXP/EPS
0068      DX = C.25
0069      IF(NX.EQ.3)DX=0.125
0070      DX = CNX**DX
0071      C
0072      NYP = NY-1
0073      TY = SIN(PI*FCY*0.5)/COS(PI*FCY*0.5)
0074      TY=TY**2
0075      IF(TY.LE.D)TY=D
0076      CNY = TY**NYP/EPS
0077      DY = 0.25
0078      IF(NY.EQ.3)DY=0.125
0079      DY = CNY**DY
0080      C      CALCULATE COEFFICIENTS
0081      C
0082      NNN=N-1
0083      DO 10 J=1,NNN
0084      DO 10 K = 1,2
0085      IF(K.EQ.2) GO TO 20
0086      CN = CNX
0087      DD = DX
0088      IF (NX.EQ.3) GO TO 22
0089      THT = 135.0*PI/180.0
0090      GO TO 23
0091      22      IF(J.EQ.1)THT=112.5*PI/180.0
0092      IF(J.EQ.2)THT=157.5*PI/180.0
0093      GO TO 23
0094      20      CN=CNY
0095      DD = DY
0096      IF (NY.EQ.3) GO TO 21
0097      THT=135.0*PI/180.0
0098      GO TO 23
0099      21      IF(J.EQ.1)THT=112.5*PI/180.0
0100      IF(J.EQ.2)THT=157.5*PI/180.0
0101      23      ALP=COS(THT)
0102      BET = SIN(THT)
0103      S1 = 1.0+ALP*DD
0104      S2 = 1.0-ALP*DD
0105      S3 = BET*DD
0106      S4=-S3
0107      P=CMPLX(S1,S3)/CMPLX(S2,S4)
0108      S1 = -2*REAL(P)
0109      S2 = (CABS(P))**2
0110      C

```

```

0111      AA = 0.25 * ( 1.0 + S1 +S2 )
0112      A(1,J,K) = AA
0113      A(2,J,K) = 2.0*AA
0114      A(3,J,K) = AA
0115      B(1,J,K) = 1.0
0116      B(2,J,K) = S1
0117 10      B(3,J,K)=S2
0118 C
0119 C      OBTAIN TWO DIMENSION FILTER
0120 C
0121      IF(NX.EQ.3)GO TO 30
0122      A(1,2,1) = 1.0
0123      B(1,2,1) = 1.0
0124 30      IF(NY.EQ.3)GO TO 31
0125      A(1,2,2) = 1.0
0126      B(1,2,2) = 1.0
0127 C
0128 31      DO 40 I = 1,3
0129      DO 40 J = 1,3
0130      DO 40 K = 1,2
0131      U(I,J,K) = A(I,K,1)*A(J,K,2)
0132 40      V(I,J,K) = B(I,K,1)*B(J,K,2)
0133      RETURN
0134      END
0135 C      *****SUBROUTINE INTRP*****
0136 C/L60
0137 C      *****SUBROUTINE INTRP*****
0138 C      *
0139 C      * THIS SUBROUTINE IS FOR INTERPOLATING POINTS IN AN ARRAY *
0140 C      *
0141 C      *****SUBROUTINE VARIABLES*****
0142 C      * AINT: STORAGE ARRAY FOR DATA POINTS *
0143 C      * Y: THE DISTANCE BETWEEN LINES OF DATA POINTS *
0144 C      * DX: INTERVAL VALUE BETWEEN INTERPOLATING POINTS *
0145 C      * NCOLS: NUMBER OF POINTS REQUIRED PER ROW IN OUTPUT *
0146 C      * ICOLS: NUMBER OF POINTS TO PER ROW IN INPUT *
0147 C      * FOP: THE OUTPUT ARRAY *
0148 C      ****
0149 C
0150 C
0151      SUBROUTINE INTRP(AINT,Y,DX,NCOLS,ICOLS,FOP)
0152      DIMENSION AINT(1), FOP(1),JMES(40)
0153      CALL CODE
0154      WRITE (JMES,150)
0155 150     FORMAT(' NOW IN INTRP')
0156      CALL TRMGN(JMES,LU,0)
0157 C
0158      IMAX=512
0159      IMXP=IMAX+1
0160      ICM1=ICOLS-1
0161      I=1
0162      M=I
0163 15      X=(I-1)*DX-(M-1)
0164      IF(X.LT.1.0) GO TO 25
0165      M=M+1
0166      IF(M.GT.ICM1) GO TO 50
0167      GO TO 15
0168 25      E=(AINT(M+1)-AINT(M))*X+AINT(M)
0169      F=(AINT(M+IMXP)-AINT(M+IMAX))*X+AINT(M+IMAX)
0170      FOP(I)=(F-E)*Y+E

```

```

0171      I=I+1
0172      IF(I.LE.NCOLS) GO TO 15
0173      IF(X.LT.1.0.AND.I.GT.NCOLS) GO TO 51
0174  C
0175  C      COMPLETE INTERPOLATION
0176  C
0177  50      FOP(NCOLS)=(AINT(ICOLS+IMAX)-AINT(ICOLS))*Y+AINT(ICOLS)
0178  51      CALL CODE
0179      WRITE(JMES,160)
0180  160      FORMAT(' NOW LEAVING INTRP')
0181      CALL TRMGN(JMES,LU,0)
0182      RETURN
0183      END

```

&LBRSZ T=00004 IS ON CRO0022 USING 00010 BLKS R=0100

```

0001  FTN4,L
0002      SUBROUTINE SPCHR (IRTCD,IRT)
0003  C
0004  C      THIS ROUTINE CHECKS FOR SPECIAL CHARACTERS IN THE INPUT DATA
0005  C
0006      IRT=5
0007      IF (IRTCD.EQ. 0B) IRT=0
0008      IF ((IRTCD.EQ.1HX) .OR. (IRTCD .EQ. 2HX ))IRT=1
0009      IF ((IRTCD.EQ.1HB) .OR. (IRTCD .EQ. 2HB ))IRT=2
0010      IF ((IRTCD.EQ.1HD) .OR. (IRTCD .EQ. 2HD ))IRT=3
0011      IF ((IRTCD.EQ.1HR) .OR. (IRTCD .EQ. 2HR ))IRT=4
0012      RETURN
0013      END
0014      SUBROUTINE CKFLD(IA,ICD,IRS)
0015  C
0016  C      SUBROUTINE TO CHECK FOR CARRIAGE RETURN OR NUMERIC VALUE
0017  C
0018      IRS=ICD+1
0019      IF (ICD .EQ. 0B) IRS=1
0020      RETURN
0021      END
0022      SUBROUTINE INFRM (IA,LU)
0023      DIMENSION IA(3)
0024      ICNWD=400B +LU
0025      CALL EXEC (2,ICNWD,IA,3)
0026      RETURN
0027      END
0028  C
0029  C
0030      SUBROUTINE XFLTR(AINT,ICOLS,F,A,B,FCX,ITME)
0031      DIMENSION B(3,2)
0032      DIMENSION F(1),AINT(1),A(3,2),WF(3),WG(3)
0033  C      IF (ITME.EQ.0) CALL BOOST(0.0,1.0,FCX,2,A,B)
0034      ITME=ITME+1
0035      A1=A(1,1)
0036      A2=A(2,1)
0037      A3=A(3,1)
0038      B2=B(2,1)
0039      B3=B(3,1)
0040  C

```

```

0041 C      INITIALIZE
0042 C
0043      IMAX=512
0044      INT=ICOLS/2-1
0045      IMXP1=IMAX+1
0046      ASTT=AINT(INT)+AIN( INT+1)+AIN( INT+2)
0047      ASTT=ASTT/3
0048      DO 10 I=1,3
0049      WF(I)=ASTT
0050 10      WG(I)=ASTT
0051 C
0052 C      START FORWARD FILTER
0053 C
0054      MM=IMXP1
0055      WF(1)=AIN( IMXP1)
0056 20      WG(1)=A1*WF(1)+A2*WF(3)+A3*WF(3)-B2*WG(2)-B3*WG(3)
0057 C
0058 C      UPDATE
0059 C
0060      AINT(MM)=WG(1)
0061      WG(3)=WG(2)
0062      WG(2)=WG(1)
0063      WF(3)=WF(2)
0064      WF(2)=WF(1)
0065      MM=MM+1
0066      IF (MM.GT. ICOLS) GO TO 30
0067      WF(1)=AIN(MM)
0068      GO TO 20
0069 C
0070 C      START REVERSE FILTER
0071 C
0072 30      ASTT=AINT(INT)
0073      DO 40 I=1,3
0074      WF(I)=ASTT
0075 40      WG(I)=ASTT
0076 C
0077      MM=IMAX+ICOLS
0078      WF(I)=AIN(MM)
0079 41      WG(1)=A1*WF(1)+A2*WF(2)+A3*WF(3)-B2*WG(2)-B3*WG(3)
0080 C
0081 C      UPDATE
0082 C
0083      AINT(MM)=WG(1)
0084      WG(3)=WG(2)
0085      WG(2)=WG(1)
0086      WF(3)=WF(2)
0087      WF(2)=WF(1)
0088      MM=MM-1
0089      IF (MM .LE. IMAX) GO TO 50
0090      WF(1)=AIN(MM)
0091      GO TO 41
0092 50      RETURN
0093 END
0094      SUBROUTINE BLANK (JMES)
0095      DIMENSION JMES(40)
0096      DO 10 I=1,40
0097 10      JMES(I)=21
0098      RETURN
0099 END
0100 END$
```

&WFINT T=00004 IS ON CRO0022 USING 00006 BLKS R=0044

```
0001  FTN4
0002      INTEGER FUNCTION WFINT(NLINE,NPIXL,PMAX,PMIN,LU)
0003  C
0004  C
0005  C  THIS SUBROUTINE IS USED IN CONJUNCTION WITH IMAGE PROCESSING
0006  C  IT CREATES AND MAINTAINS AN IMAGE WORK FILE WITH PIXEL VALUES
0007  C  STORED AS REAL NUMBERS TO PRESERVE PRECISION.
0008  C
0009  C  THIS ONE INITIALIZES THE PROCESS BY CREATING THE WORK FILE
0010  C  AND RETURNING CERTAIN PERTINENT INFO TO CALLER.  IT SHOULD
0011  C  ONLY BE CALLED ONCE BY EACH CALLER.  THE OTHER TWO ARE
0012  C  READL, WHICH READS A PARTICULAR LINE AND RITEL WHICH WRITES
0013  C  A PARTICULAR LINE.
0014  C
0015  C      LU      = INTERACTIVE TERMINAL LU
0016  C
0017  C      NLINE = # LINES IN IMAGE
0018  C      NPIXL = # PIXELS/LINE
0019  C      PMAX  = MAXIMUM PIXEL INTENSITY IN IMAGE (REAL)
0020  C      PMIN  = MINIMUM PIXEL INTENSITY IN IMAGE (REAL)
0021  C
0022  C
0023  C      DIMENSION IDCBI(144),IRTN(5),IB(6)
0024  C
0025  C      EQUIVALENCE (IB2,IB(2)),(IB(3),RMAX),(IB(5),RMIN)
0026  C
0027  C
0028  C
0029  C  SCHEDULE BUILD WORK FILE PROGRAM
0030  C
0031  C      CALL EXEC(23,6HBLDWF ,LU)
0032  C
0033  C  GET RETURNED PARAMETERS
0034  C
0035  C      CALL RMPAR(IRTN)
0036  C      WFINT = IRTN
0037  C      IF (IRTN .LT. 0 ) RETURN
0038  C
0039  C  GET MAX MIN DATA
0040  C
0041  C      CALL OPEN(IDCB1,IERR,6HWF0000)
0042  C      IF (IERR .LT. 0 ) GO TO 100
0043  C
0044  C      CALL READF(IDCB1,IERR,IB,6)
0045  C      IF (IERR .LT. 0 ) GO TO 100
0046  C      NLINE = IB
0047  C      NPIXL = IB2
0048  C      PMAX  = RMAX
0049  C      PMIN  = RMIN
0050  C      CALL CLOSE(IDCB1)
0051  C      WFINT = 0
0052  C      RETURN
0053  C
0054  100  WFINT = IERR
0055  C      CALL CLOSE(IDCB1)
0056  C      END
0057  C
```

```
      0058 C
0059 C READ LINE FROM WORK FILE SUBROUTINE
0060 C
0061 C
0062     INTEGER FUNCTION READL(LINE,IPIXL,JPIXL,RBUF)
0063 C
0064     COMMON /CBLK/IDCB(528),TBUF(512),IFLAG
0065 C
0066     DIMENSION RBUF(512)
0067 C
0068 C
0069 C CHECK IF FILE OPEN
0070 C
0071     IF (IFLAG .EQ. 1) GO TO 100
0072 C
0073 C MUST OPEN FILE
0074 C
0075     CALL OPEN(IDCB,IERR,6HWF0000,2,0,0,528)
0076     IF (IERR .LT. 0) GO TO 999
0077     IFLAG = 1
0078 C
0079 C FILE OPENED--READ APPROPRIATE LINE
0080 C
0081 100    CALL READF(IDCB,IERR,TBUF,1024,LEN,LINE+2)
0082     IF (IERR .LT. 0) GO TO 999
0083 C
0084 C POSITION DATA IN BUFFER
0085 C
0086     ISTEP = 1
0087     IF (IPIXL .GT. JPIXL) ISTEP = -1
0088 C
0089     J = 1
0090     DO 110 I=IPIXL+1,JPIXL+1,ISTEP
0091     RBUF(J) = TBUF(I)
0092 110    J = J+1
0093     READL = 0
0094     RETURN
0095 C
0096 C ERROR
0097 C
0098 999    READL = IERR
0099     END
0100 C
0101 C
0102 C WRITE WORK FILE SUBROUTINE
0103 C
0104 C
0105     INTEGER FUNCTION RITEL(LINE,IPIXL,JPIXL,RBUF)
0106 C
0107     COMMON /CBLK/IDCB(528),TBUF(512),IFLAG
0108 C
0109     DIMENSION RBUF(512)
0110 C
0111 C CHECK IF FILE OPENED
0112 C
0113     IF (IFLAG .EQ. 1) GO TO 100
0114 C
```

0115 C MUST OPEN FILE
0116 C
0117 CALL OPEN(IDCB,IERR,6HWF0000,2,0,0,528)
0118 IF (IERR .LT. 0) GO TO 999
0119 IFLAG = 1
0120 C
0121 C FILE OPENED--WRITE APPROPRIATE LINE
0122 C
0123 100 CALL READF(IDCB,IERR,TBUF,1024,LEN,LINE+2)
0124 IF (IERR .LT. 0) GO TO 999
0125 C
0126 ISTEP = 1
0127 IF (IPIXL .GT. JPIXL) ISTEP = -1
0128 J = 1
0129 DO 110 I=IPIXL+1,JPIXL+1,ISTEP
0130 TBUF(I) = RBUF(J)
110 J = J+1
0132 C
0133 CALL WRITF(IDCB,IERR,TBUF,0,LINE+2)
0134 IF (IERR .LT. 0) GO TO 999
0135 C
0136 RITEL = 0
0137 RETURN
0138 C
0139 C ERROR RETURN
0140 C
0141 999 RITEL = IERR
0142 END
0143 C
0144 C
0145 C BLOCK DATA SUBROGRAM
0146 C
0147 C
0148 BLOCK DATA
0149 C
0150 COMMON /CBLK/IDCB(528),TBUF(512),IFLAG
0151 C
0152 DATA IFLAG/0/
0153 C
0154 END
0155 C
0156 C
0157 C CLOSE WORK FILE SUBROUTINE
0158 C
0159 C
0160 SUBROUTINE CLSWF(NLINE,NPIXL,PMAX,PMIN) ORIGINAL PAGE IS
0161 C OFF PWR QUALITY
0162 COMMON /CRLK/ IDCL(528)
0163 C
0164 DIMENSION IB(6)
0165 C
0166 EQUIVALENCE (IB2,IB(2)),(IB(3),RMAX),(IB(5),RMIN)
0167 C
0168 C
0169 C THIS ROUTINE IS USED TO CLOSE THE WORK FILE
0170 C
0171 C WRITE DATA ON WORK FILE
0172 C
0173 IB = NLINE
0174 IB2 = NPIXL
0175 RMAX = PMAX
0176 RMIN = PMIN
0177 C
0178 CALL WRITF(IDCB,IERR,IB,6,1)
0179 CALL CLOSE(IDCB)

C - 2

&DINTF T=00004 IS ON CR00022 USING 00009 BLKS R=0087

```
0001  FTN4,Q,T,C
0002      PROGRAM DINTP
0003  C
0004  C      THIS PROGRAM IS USED CHANGE THE PHYSICAL SIZE OF AN IMAGE
0005  C
0006  C      WRITTEN BY WINSER E. ALEXANDER
0007  C
0008      DIMENSION AINT(1024),F(512),IPRAM(5),DIRC(515),INM(2)
0009      EQUIVALENCE (F(1),DIRC(4)),(DIRC(1),INM(1))
0010      EQUIVALENCE (INM(1),NROW),(INM(2),ICOLS),(DIRC(2),AMAX)
0011      EQUIVALENCE (DIRC(3),AMIN)
0012  C
0013  C      INPUT PARAMETERS (CALL RMPAR)
0014  C      IPRAM(1)= LOGICAL UNIT FOR INTERACTIVE DEVICE
0015  C      IPRAM(2) = NUMBER OF DESIRED ROWS IN OUTPUT IMAGE
0016  C      IPRAM(3) = NUMBER OF DESIRED COLUMNS IN OUTPUT IMAGE
0017  C
0018  C      IMAGE TO BE USED IS ASSUMED TO BE IN IMAGE WORK FILE (WF0000
0019  C
0020      CALL RMPAR(IPRAM)
0021      LU=IPRAM(1)
0022      IF(LU.LE.0) LU=1
0023      NNEW=IPRAM(2)
0024      NCOLS=IPRAM(3)
0025      NCM1=NCOLS-1
0026      IMX=512
0027      IMXP1=IMX+1
0028  C
0029  C      OBTAIN PARAMETERS FROM CURRENT IMAGE
0030  C
0031      IGET=READL(-1,0,511,DIRC)
0032      ICM1=ICOLS-1
0033      IF(IGET.LT.0) GO TO 999
0034  C
0035  C      INTERPOLATE IMAGE
0036  C
0037      DY=FLOAT(NROW)/FLOAT(NNEW)
0038      DX=FLOAT(ICOLS)/FLOAT(NCOLS)
0039      IFLT=0
0040      IF(DY.GT.1.0) STOP 111
0041      IF(DX.LT.0.0) IFLT=1
0042      IFR=0
0043      IFE=0
0044  C
```

```
0045 C      INITIALIZE ARRAYS
0046 C
0047      IGET=READL(0,IFE,ICOLS-1,AINT(MXP1))
0048      IF(IGET.LT.0) GO TO 999
0049      IGET=READL(1,IFE,ICM1,AINT(1))
0050      IF(IGET.LT.0) GO TO 999
0051 C
0052      MCNT=2
0053      MORG=NROW
0054      DO 100 KK=NNEW,1,-1
0055 C
0056 C      COMPUTE Y
0057 C
0058      20 Y=(NNEW-KK)*DY-(NROW-MORG)
0059      IF(Y.LT.1.0) GO TO 50
0060 C
0061 C      BRING IN NEW ROW
0062 C
0063      CALL MOVE(AINT,ICOLS,IMXP1)
0064 C
0065      MCNT=MCNT+1
0066      IGET=0
0067      IF(MCNT.GT.NROW) IGET=-150
0068      IF(IGET.LT.0) GO TO 999
0069      IGET=READL(MCNT,IFE,ICM1,AINT)
0070      IF(IGET.LT.0) GO TO 999
0071      MORG=MORG-1
0072 C
0073 C      RECOMPUTE Y
0074 C
0075      GO TO 20
0076 C
0077 C      INTERPOLATE FOR NEW ROW
0078 C
0079      50 CALL INTRP(AINT,Y,DX,NCOLS,ICOLS,F)
0080 C
0081 C      OUTPUT CURRENT ROW
0082 C
0083      100 CALL RITEL(KK-1,0,NCM1,F)
0084 C
0085 C      NOTE THAT WORK FILE IS NOT CLOSED BY THIS PROGRAM
0086 C
0087 C      INSERT PARAMETERS IN WORK FILE
0088 C
0089      NROW=NNEW
0090      ICOLS=NCOLS
0091      CALL RITEL(-1,0,ICM1,DIRC)
0092      999 CONTINUE
0093 C
0094 C      ERROR PROCESSING
0095 C
0096      WRITE(LU,1000) IGET
0097      1000 FORMAT(" ERROR CODE = ",1I5)
0098      CALL EXEC(6)
0099      END
0100 C
```

```

0101 C
0102 SUBROUTINE MOVE(A INT, ICOLS, IMXP1)
0103 C
0104 C      THIS SUBROUTINE MOVES ICOLS ELEMENTS IN ARRAY AINT FROM
0105 C      A START POINT OF 1 TO A START POINT OF IMXP1
0106 C
0107      DIMENSION AINT(1)
0108 C
0109      DO 10 I=1,ICOLS
0110 10 AINT(IMXP1+I) = AINT(I)
0111      RETURN
0112      END
0113      ENDS

```

&WTAPE T=00004 IS ON CR00022 USING 00012 BLKS R=0127

```

0001 FTN4,Q,C,T
0002      PROGRAM WTAPE
0003 C
0004 C      THIS PROGRAM FROMS A PART OF THE IMAGE PROCESSING SYSTEM
0005 C
0006 C      IT IS USED TO STORE AN IMAGE ON TAPE AND THEN PURGE FROM DIS
0007 C
0008 C      THE IMAGE INVENTOPRY FILE IS UPDATED TO SHOW THAT THE IMAGE
0009 C      TAPE
0010 C
0011 C      WRITTEN BY WINSER E. ALEXANDER
0012 C
0013      DIMENSION IDCBL(272),IDCB2(528),IMAGE(6),IPRAM(5),JNAME(3)
0014      DIMENSION IDATA(512),ISIZE(2),IRTN(5),IBUF(15)
0015 C
0016      EQUIVALENCE (IBUF(12),ILOC),(IBUF(13),JNAME),(IBUF(7),NLINE)
0017      EQUIVALENCE (IBUF(8),NPIXL),(IBUF(9),IPMIN),(IBUF(10),IPMAX)
0018 C
0019 C      GET INPUT PARAMETERS
0020 C
0021      CALL RMPAR(IPRAM)
0022      LU = IPRAM(1)
0023      IF(LU.LE.0) LU=1
0024 C
0025      LU2 = 8
0026 C
0027 C      POSITION TAPE
0028 C
0029      WRITE(LU,45)
0030      READ(LU,46) IOPT
0031      IF (IOPT .NE. 2HGO) GO TO 1000
0032      WRITE(LU,51)
0033      READ(LU,*) IFNUM
0034 C
0035 C      SPACE TO FILE POSITION
0036 C
0037      CALL EXEC(3,400B+LU2)
0038      IF (IFNUM .LE. 0) GO TO 1000
0039      IF (IFNUM .EQ. 1) GO TO 5
0040 C

```

```
0041      DO 55 I=1,IFNUM-1
0042      CALL EXEC(3,1300B+LU2)
0043 55      CONTINUE
0044 C
0045 C      GET IMAGE NAME FROM USER
0046 C
0047 C
0048 5      WRITE(LU,10)
0049 10 FORMAT(" ENTER IMAGE NAME (12 CHARACTERS /E TO EXIT)? _")
0050      READ(LU,20) IMAGE
0051 20 FORMAT(6A2)
0052      IF (IMAGE .EQ. 2H/E) GO TO 1001
0053 C
0054 C      OPEN DIRECTORY FILE
0055 C
0056 30 CALL OPEN(IDCB1,IERR,6HIMCIRC,1,2HIM,23,272)
0057      IF(IERR.LT.0) GO TO 999
0058 C
0059 C      FIND IMAGE FILE
0060 C
0061 40 CALL READF(IDCB1,IERR,IBUF,15,LEN)
0062      IF(LEN.NE.-1) GO TO 35
0063      WRITE(LU,36)
0064 36      FORMAT("IMAGE NOT FOUND")
0065      GO TO 5
0066 35      IF(IERR.LT.0) GO TO 999
0067 C
0068      IF(ICMPW(IMAGE,IBUF,6).NE.0) GO TO 40
0069 C
0070 C      IMAGE FOUND
0071 C
0072 C      CLOSE DIRECTORY FILE AND OPEN IMAGE FILE
0073 C
0074      CALL CLOSE(IDCB1)
0075 C
0076      CALL OPEN(IDCB2,IERR,JNAME,1,2HIM,23,528)
0077      IF(IERR.LT.0) GO TO 999
0078 C
0079 C      CHECK FOR TAPE ON TRANSPORT
0080 C
0081 45      FORMAT(" PUT TAPE ON TRANSPORT & PUT TAPE UNIT ON LINE.")
0082      /*" ENTER -GO- WHEN READY"*/
0083 46 FORMAT(1A2)
0084 C
0085 C
0086      WRITE(LU,48) IPMAX,IPMIN
0087 48 FORMAT(" MAXIMUM VALUE = ",1I8,". MINIMUM = ",1I8)
0088 C
0089 C      IF(IPMAX.LE.255) IMAGE WILL BE PACKED FOR OUTPUT (8 BIT IMAG
0090 C
0091      ITYPE = 15
0092      IF(IPMAX.LE.255.AND.IPMIN.GE.0) ITYPE = 8
0093 C
0094 C
0095 51      FORMAT("FILE #? _")
0096 C
0097 C
```

```
0098 C
0099 C OUTPUT DATA TO TAPE
0100 C
0101 60 DO 80 I =1,512
0102 CALL FILL(IDATA,0,512)
0103 IERR =0
0104 IF (I .LE. NLINE) CALL READF(IDCB2,IERR,IData,512)
0105 IF (IERR .LT. 0) GO TO 999
0106 C
0107 NOUT = 512
0108 IF (ITYPE .EQ. 15) GO TO 70
0109 C
0110 C PACK DATA
0111 C
0112 DO 65 J=1,512,2
0113 CALL ROT8(IDATA(J),ITEMP)
0114 65 IData(J) = IOR(ITEMP,IData(J+1))
0115 NOUT = 256
0116 C
0117 C WRITE DATA
0118 C
0119 70 CALL EXEC(2,LU2,IData,NOUT)
0120 80 CONTINUE
0121 C
0122 CALL EXEC(3,100B+LU2)
0123 CALL CLOSE(IDCB2)
0124 GO TO 5
0125 C
0126 C FILE ERROR
0127 C
0128 999 WRITE(LU,996) IERR
0129 996 FORMAT(" FILE ERROR = ",1I4)
0130 CALL CLOSE(IDCB1)
0131 CALL CLOSE(IDCB2)
0132 1001 CALL EXEC(3,400B+LU2)
0133 C
0134 1000 CONTINUE
0135 END
0136 C
0137 C
0138 SUBROUTINE FILL(IARAY,ICHAR,NUM)
0139 C
0140 C THIS SUBROUTINE IS USED TO FILL THE ARRAY "IARAY" WITH
0141 C THE CHARACTER "ICHAR"
0142 C DIMENSION IARAY(NUM)
0143 C
0144 DO 10 I=1,NUM
0145 10 IARAY(I)=ICHAR
0146 RETURN
0147 END
0148 END$
```

```

0001  FTN4,L
0002      PROGRAM PLOTV
0003      DIMENSION LU(5)
0004      INTEGER IDCB(144),BUFF( 4 ), NAME(3)
0005      DATA NAME/2HDA,2HTA,2H1 /
0006  C
0007  C GET LU
0008  C
0009      CALL RMPAR(LU)
0010  C
0011      CALL INITA(0)
0012      CALL OPEN(IDCB,IERR,NAME)
0013      IF (IERR .GE. 0) GO TO 20
0014      WRITE(LU,10) IERR
0015  10      FORMAT ("OPEN ERROR",F5.0)
0016      STOP
0017  C20      CONTINUE
0018  20      CALL READF(IDCB,IERR,BUFF,4,IERR)
0019      IF (IERR .GE. 0) GOTO 40
0020      WRITE(LU,30) IERR
0021  30      FORMAT("READ ERROR",F5.0)
0022      GO TO 55
0023  40      CONTINUE
0024      CALL DVECT(BUFF,BUFF(2),BUFF(3),BUFF(4),LU)
0025  50      GO TO 20
0026  55      STOP
0027      END
0028      SUBROUTINE DVECT(IX1,IY1,IX2,IY2,LU)
0029      DIMENSION IBUFF(5)
0030  C
0031  C      DRAWS A VECTOR BETWEEN X1,Y1 AND X2,Y2
0032  C
0033      SCAL =255./1024.
0034      IBUFF1  =(SCAL*IX1+0.5)+128
0035      IBUFF2  =(SCAL*IY1+0.5)
0036      IBUFF3  =(SCAL*IX2+0.5)+128
0037      IBUFF4  =(SCAL*IY2+0.5)
0038      IBUFF1 = LAND(IBUFF1,777B)
0039      IBUFF2 = LAND(IBUFF2,377B)
0040      IBUFF3 = LAND(IBUFF3,777B)
0041      IBUFF4 = LAND(IBUFF4,377B)
0042      IBUFF(1) = IBUFF1 + 44000B
0043      IBUFF(2) = IBUFF2 + 64000B
0044      IBUFF(3) == IBUFF1 + IBUFF3 + 50000B + 512
0045      IBUFF(4) == IBUFF2 + IBUFF4 + 72000B + 256
0046  C
0047      CALL DRIVR(2,IBUFF,4)
0048  C
0049      RETURN
0050      END
0051      SUBROUTINE INITA(IBACK)
0052      DIMENSION INIT(6)
0053      DATA INIT/30000B,100377B,10377B,24021B,26000B/
0054  C
0055  C      IBACK = 1 FOR REVERSE BACKGROUND
0056  C      INITIALIZE
0057  C
0058      IF(IBACK .EQ. 1) INIT(4)=24221B
0059      CALL DRIVR(2,INIT,5)
0060      RETURN
0061      END
0062  $

```

&DPLAM T=00004 IS ON CRO0022 USING 00022 BLKS R=0210

```
0001  FTN4,L
0002      PROGRAM DPLAM
0003  C
0004  C THIS PROGRAM DISPLAYS THE FILTER CHARACTERISTICS
0005  C
0006  C
0007      COMMON/CNT/XM(30,30)
0008      COMMON/WORK/WO(130)
0009      COMMON/QDCAZ/IQ(40)
0010      INTEGER BUFF
0011      COMMON/ /IDCB(144),BUFF(10)
0012      DIMENSION IBUF(80),ILU(5),A(25),B(25),AA(5,5),BB(5,5)
0013      DIMENSION XXX(31),YYY(31),XYP(31,2),LXY(15,3),AR(60)
0014      DIMENSION XERR(31),CZ( 9),IREG(2),U(3,3,2),V(3,3,2)
0015      COMPLEX HA,HB,Z(25)
0016      EQUIVALENCE (AA(1,1),XM(1,1)),(BB(1,1),XYP(1,1))
0017      EQUIVALENCE (IREG(1),REG)
0018      EQUIVALENCE (IBUF(1),U(1,1,1)),(IBUF(41),V(1,1,1))
0019  C
0020      CALL RMPAR(ILU)
0021      LU=ILU(1)
0022      MN=ILU(2)
0023      AMAX = 0.0
0024      AMIN =1000.0
0025      MDIM = 30
0026      NDIM = 30
0027  C
0028  C
0029  C GET FILTER COEFF'S
0030      CALL EXEC(14,1,IBUF,80)
0031  C
0032      IF(MN.EQ.3) GO TO 100
0033      MNL=9
0034      DO 10 J=1,3
0035      DO 10 K=1,3
0036      II=J+(K-1)*3
0037      A(II)=U(J,K,1)
0038      10  B(II)=V(J,K,1)
0039      GO TO 101
0040      100 DO 103 I=1,5
0041      DO 103 J=1,5
0042      AA(I,J)=0.0
0043      103 BB(I,J)=0.0
0044      DO 102 I=1,3
0045      DO 102 J=1,3
0046      DO 102 K=1,3
0047      DO 102 L=1,3
0048      IK=I+K-1
0049      JL=J+L-1
0050      AA(IK,JL)=AA(IK,JL)+U(I,J,1)*U(K,L,2)
0051      102 BB(IK,JL)=BB(IK,JL)+V(I,J,1)*V(K,L,2)
0052      DO 11 J=1,5
0053      DO 11 K=1,5
0054      II=J+(K-1)*5
0055      A(II)=AA(J,K)
0056      11  B(II)=BB(J,K)
0057      MNL=25
```

```

0058 101  WRITE(LU,101)
0059 1011 FORMAT(21H COEFFICIENT MATRICES,/ )
0060          WRITE(LU,105) (A(I),I=1,25)
0061          WRITE(LU,105)(B(I),I=1,25)
0062      105 FORMAT(5(1H ,5E10.2//))
0063 C
0064 C      COMPUTE THE CENTER OF OUTPUT ARRAY
0065 C
0066      WRITE(LU,12)
0067      12  FORMAT(" ENTER MX FOR HORIZONTAL FREQUENCIES"/)
0068          READ(LU,13)MX
0069      13  FORMAT(1I2)
0070          WRITE(LU,14)
0071      14  FORMAT(" ENTER MY FOR VERTICAL FREQUENCIES"/)
0072          READ(LU,13) MY
0073      203  MXC=MX/2
0074          MXT=2*MXC
0075          NX=0
0076          IF(MXT.NE.MX) NX=1
0077          MYC=MY/2
0078          MYT=2*MYC
0079          NY=0
0080          IF(MYT.NE.MY) NY=1
0081          MXN=MXC+1
0082          MYN=MYC+1
0083      WRITE(LU,301)
0084      300 FORMAT(" COMPUTE SQUARED MAGNITUDE ")
0085      301 FORMAT(" INITIALIZE ARRAY")
0086 C
0087 C      COMPUTE SQUARED MAGNITUDE CHARACTERISTIC
0088 C
0089          MX=MXT+NX
0090          MY=MYT+NY
0091          FCX=2.0/FLOAT(MX)
0092          FCY=2.0/FLOAT(MY)
0093          IF(MX.LE.101.AND.MY.LE.61) GO TO 204
0094          IF(MX.LE.101) GO TO 202
0095          MX=101
0096      WRITE(LU,200)
0097      200  FORMAT(" SIZE OF ARRAY WAS REDUCED TO 31 FOR HORIZONTAL ")
0098      202  IF(MY.LE.61) GO TO 203
0099          MY=61
0100      WRITE(LU,201)
0101      GO TO 203
0102      201  FORMAT(" SIZE OF ARRAY WAS REDUCED TO 31 FOR VERTICAL ")
0103      204 WRITE(LU,300)
0104          DO 20 I=1,MX+1
0105          DO 20 J=1,MY+1
0106          XF=FCX*(I-MXC-1)
0107          XXX(I)=XF
0108          YF=FCY*(J-MYC-1)
0109          YYY(J)=YF
0110          CALL ZWC(Z,XF,YF,MN)
0111          HA=CMPLX(0.0,0.0)
0112          HB=HA
0113          DO 21 K=1,MNL
0114          HA=HA+A(K)*Z(K)
0115          HB=HB+B(K)*Z(K)

```

```

0116 21      CONTINUE
0117      XA=CABS(HA)
0118      XB=CABS(HB)
0119      IF(XB.LE.1.0E-20) XB=1.0E-20
0120      XA=XA/XB
0121      XM(I,J)=XA**2
0122      IF(XM(I,J).LT.AMIN) AMIN=XM(I,J)
0123      IF(XM(I,J).GT.AMAX) AMAX=XM(I,J)
0124      20 CONTINUE
0125      WRITE(LU,302)AMAX,AMIN
0126      302 FORMAT(" AMAX = ",1E10.2,3X,"AMIN = ",1E10.2/)
0127      C
0128      C SQUARED MAGNITUDE NORMALIZED
0129      C
0130      C
0131      C      OBTAIN W=1 PLOT FROM ARRAY
0132      C
0133      DO 22 I=1,MXN
0134      XYP(I,2)=XM(I+MXC,MYN)
0135      XERR(I)=XYP(I,2)
0136      22 XYP(I,1)=FCX*(I-1)
0137      WRITE(LU,306)(XYP(I,1),XYP(I,2),I=1,MXN)
0138      306 FORMAT(//1H ,6(1E10.2,3X)/)
0139      XL=0.0
0140      XU=1.0
0141      MC=2
0142      C
0143      C      CALCULATE Z=W PLOT
0144      C
0145      X=(MXN)**2+(MYN)**2
0146      FCX=0.7071*FCX
0147      NUM=SQRT(X)+1
0148      DO 30 I=1,MXN
0149      XF=FCX*(I-1)
0150      CALL ZWC(Z,XF,XF,MN)
0151      HA=CMPLX(0.0,0.0)
0152      HB=HA
0153      DO 31 K=1,MNL
0154      HA=HA+A(K)*Z(K)
0155      31      HB=HB+B(K)*Z(K)
0156      XA=CABS(HA)
0157      XB=CABS(HB)
0158      IF(XB.LE.1.0E-20)XB=1.0E-20
0159      XA=XA/XB
0160      XYP(I,2)=XA**2
0161      30      XYP(I,1)=XF*1.414
0162      WRITE(LU,306)(XYP(I,1),XYP(I,2),I=1,MXN)
0163      C

```

```

0164 C COMPUTE ERROR FUNCTION
0165 C
0166 ERR=0.0
0167 DO 350 J=1,MDIM
0168 350 ERR=ERR+((XERR(J)-XYP(J,2))/AMAX)**2
0169 WRITE(LU,360) ERR
0170 360 FORMAT(" RELATIVE ERROR = ",1E15.7/)
0171 C COMPUTE CONTOURS
0172 C
0173 C
0174 C PLOT IMAGE OF TRANSFER FUNCTION
0175 C
0176 CALL CONTR(XXX,YYY,AMAX,AMIN,MX+1,MY+1,LU)
0177 C
0178 4443 STOP
0179 END
0180 SUBROUTINE ZWC(Z,XF,YF,MN)
0181 C
0182 C THIS SUBROUTINE COMPUTES COMPLEX
0183 C VALUES FOR Z**I*W**J FOR
0184 C ZW TRANSFORM AND PLACES RESULTS
0185 C IN ONE DIMENSIONAL ARRAY Z
0186 C XF=HORIZONTAL RELATIVE FREQUENCY
0187 C YF=VERTICAL RELATIVE FREQUENCY
0188 C
0189 COMPLEX Z(25),R,S
0190 IF(ABS(XF).EQ.1.0) XF = 0.99
0191 IF(ABS(YF).EQ.1.0) YF = 0.99
0192 PI=3.1415926
0193 RX=COS(PI*XF)
0194 RY=SIN(PI*XF)
0195 SX=COS(PI*YF)
0196 SY=SIN(PI*YF)
0197 R=CMPLX(RX,RY)
0198 S=CMPLX(SX,SY)
0199 IF(MN.GE.3) GO TO 20
0200 DO 10 J=1,3
0201 DO 10 K=1,3
0202 I=J+(K-1)*3
0203 10 Z(I)=S***(J-1)*R***(K-1)
0204 GO TO 22
0205 20 DO 21 J=1,5
0206 DO 21 K=1,5
0207 I=J+(K-1)*5
0208 21 Z(I)=S***(J-1)*R***(K-1)
0209 22 RETURN
0210 END
0211 BLOCK DATA WORK
0212 COMMON/WORK/W0(130)
0213 COMMON/CNT/XM(900)
0214 COMMON/QDCAZ/IQ(40)
0215 END
0216 $

```

&DPLA1 T=00004 IS ON CRO0022 USING 00052 BLKS R=0437

```

0001  FTN4,L
0002      SUBROUTINE CONTR(XXX,YYY,AMAX,AMIN,MX,MY,LU)
0003      COMMON/CNT/XM(30,30)
0004      DIMENSION XXX(MX),YYY(MY),CZ(9),ISIZE(2)
0005      INTEGER BUFF,NAME9(3)
0006      COMMON /IDCB(144),BUFF(4)
0007      DATA NAME9/2HDA,2HTA,2H1 /
0008      WRITE(LU,100)
0009 100      FORMAT(" SELECT TYPE OF FILTER PLOT "/
0010      1" 1. CONTOUR"/" 2. PERSPECTIVE"/)
0011      READ(LU,*) IFLAG
0012  C
0013  C      GENERATE CZ
0014  C
0015      CZ(1)=-1.
0016      DO 3 K=2,9
0017      CZ(K)=CZ(K-1)+.25
0018 3  CONTINUE
0019  C
0020  C CREATE A PLOT DATA FILE
0021  C
0022      ITYPE=3
0023      ISIZE(1)=96
0024      CALL PURGE(IDCB,IERR,NAME9)
0025      IF(IERR .LT. 0) WRITE(LU,101) IERR
0026      CALL CREAT(IDCB,IERR,NAME9,ISIZE,ITYPE)
0027      IF(IERR .GE. 0) GO TO 201
0028      WRITE(LU,101) IERR
0029 101      FORMAT("CREATE ERROR",F5.0)
0030      STOP
0031  C
0032  C      DO 3-D PLOTS
0033  C
0034 201      IF (IFLAG.EQ.1) GO TO 10
0035      IF (IFLAG.EQ.2) GO TO 20
0036 20      CONTINUE
0037      CALL SET3D(1.,-1.,1.,-1.,AMAX,AMIN,2,0,.5,.5)
0038      CALL PLT3D(XXX,YYY,XM,30,MX,MY,LU)
0039      IF(IFLAG.EQ.2) GOTO 30
0040  C
0041  C      DO ISOGRAMS
0042  C
0043 10      CONTINUE
0044      DO 11 I=1,MX
0045      DO 11 J=1,MY
0046      XM(I,J)=XM(I,J)/AMAX
0047 11      CONTINUE
0048      CALL SET2D(1.,-1.,1.,-1.,3,0,1.)
0049      CALL PLT2D(XXX,YYY,XM,30,MX,MY,CZ,9,LU)
0050 30      CONTINUE
0051      CALL CLOSE(IDCB)
0052      RETURN
0053      END

```

```

0054      SUBROUTINE SET2D(ALPMAX,ALPMIN,BETMAX,BETMIN,IORGN,IALPCL,AL
0055      COMMON/ QDCAZ/ IXYXYB(4,4),XZ,AX,BX,YZ,AY,BY
0056      DATA XCNTR, YCNTR, EL/512., 512., 1000./
0057      XZ=XCNTR
0058      YZ= YCNTR
0059      IF( ALTOBL-1.)6,7,8
0060      6  CONTINUE
0061      ELALP=ALTOBL*EL
0062      ELBET=EL
0063      GO TO 9
0064      7  CONTINUE
0065      ELALP=EL
0066      ELBET=EL
0067      GO TO 9
0068      8  CONTINUE
0069      ELALP=EL
0070      ELBET=EL/ALTOBL
0071      9  CONTINUE
0072      IF (IORGN.EQ.1) GO TO 1
0073      IF (IORGN.EQ.2) GO TO 2
0074      IF (IORGN.EQ.3) GO TO 3
0075      GO TO 4
0076      1  CONTINUE
0077      IF (IALPCL.EQ.1) GO TO 10
0078      BX=0.
0079      AY=0.
0080      AX=-ELALP/(ALPMAX-ALPMIN)
0081      BY=-ELBET/(BETMAX-BETMIN)
0082      XZ=XZ+.5*ELALP
0083      YZ=YZ+.5*ELBET
0084      GO TO 5
0085      10 CONTINUE
0086      AX=0.
0087      BY=0.
0088      BX=-ELBET/(BETMAX-BETMIN)
0089      AY=-ELALP/(ALPMAX-ALPMIN)
0090      XZ=XZ+.5*ELBET
0091      YZ=YZ+.5*ELALP
0092      GO TO 5
0093      2  CONTINUE
0094      IF(IALPCL.EQ.1) GO TO 20
0095      AX=0.
0096      BY=0.
0097      AY=ELALP/(ALPMAX-ALPMIN)
0098      BX=-ELBET/(BETMAX-BETMIN)
0099      XZ=XZ+.5*ELBET
0100      YZ=YZ-.5*ELALP
0101      GO TO 5
0102      20 CONTINUE
0103      AY=0.
0104      BX=0.
0105      AX=-ELALP/(ALPMAX-ALPMIN)
0106      BY=ELBET/(BETMAX-BETMIN)
0107      XZ=XZ+.5*ELALP
0108      YZ=YZ-.5*ELBET
0109      GO TO 5
0110      3  CONTINUE

```

```

0111      IF (IALPCL.EQ.1) GO TO 30
0112      AY=0.
0113      BX=0.
0114      AX=ELALP/(ALPMAX-ALPMIN)
0115      BY=ELBET/(BETMAX-BETMIN)
0116      XZ=XZ-.5*ELALP
0117      YZ=YZ-.5*ELBET
0118      GO TO 5
0119      30 CONTINUE
0120      AX=0.
0121      BY=0.
0122      AY=ELALP/(ALPMAX-ALPMIN)
0123      BX=ELBET/(BETMAX-BETMIN)
0124      XZ=XZ-.5*ELBET
0125      YZ=YZ-.5*ELALP
0126      GO TO 5
0127      * CONTINUE
0128      IF(IALPCL.EQ.1) GO TO 40
0129      AX=0.
0130      BY=0.
0131      AY=-ELALP/(ALPMAX-ALPMIN)
0132      BX=ELBET/(BETMAX-BETMIN)
0133      XZ=XZ-.5*ELBET
0134      YZ=YZ+.5*ELALP
0135      GO TO 5
0136      40 CONTINUE
0137      AY=0.
0138      BX=0.
0139      AX=ELALP/(ALPMAX-ALPMIN)
0140      BY=-ELBET/(BETMAX-BETMIN)
0141      XZ=XZ-.5*ELALP
0142      YZ=YZ+.5*ELBET
0143      5 CONTINUE
0144      XZ=XZ-AX*ALPMIN-BX*BETMIN
0145      YZ=YZ-AY*ALPMIN-BY*BETMIN
0146      IXYXYE(1,1)=IFIX(XZ+AX*ALPMIN+BX*BETMIN)
0147      IXYXYB(2,1)=IFIX(YZ+AY*ALPMIN+BY*BETMIN)
0148      IXYXYB(1,2)=IFIX(XZ+AX*ALPMIN+BX*BETMAX)
0149      IXYXYB(2,2)=IFIX(YZ+AY*ALPMIN+BY*BETMAX)
0150      IXYXYB(1,3)=IFIX(XZ+AX*ALPMAX+BX*BETMAX)
0151      IXYXYB(2,3)=IFIX(YZ+AY*ALPMAX+BY*BETMAX)
0152      IXYXYB(1,4)=IFIX(XZ+AX*ALPMAX+BX*BETMIN)
0153      IXYXYB(2,4)=IFIX(YZ+AY*ALPMAX+BY*BETMIN)
0154      IXYXYB(3,1)=IXYXYB(1,2)
0155      IXYXYB(4,1)=IXYXYB(2,2)
0156      IXYXYB(3,2)=IXYXYB(1,3)
0157      IXYXYB(4,2)=IXYXYB(2,3)
0158      IXYXYB(3,3)=IXYXYB(1,4)
0159      IXYXYB(4,3)=IXYXYB(2,4)
0160      IXYXYB(3,4)=IXYXYB(1,1)
0161      IXYXYB(4,4)=IXYXYB(2,1)
0162      RETURN
0163      END

```

```

0164 C ****
0165 C
0166 C
0167      SUBROUTINE PLT2D(ALPHA,BETA,GAMMA,IMDN,IALPHA,JBETA,C,NUMC
0168      1,IFILE,LU)
0169      COMMON/QDCAZ /IXYXYB(4,4),XZ,AX,BX,YZ,AY,BY
0170      DIMENSION ALPHA(1),BETA(1),GAMMA(IMDN,1),C(1)
0171      INTEGER BUFF(4),NAME9(3)
0172      COMMON IDCB(144)
0173      COMMON/WORK/IXIY(2,62),JXJY(2,62)
0174      DATA NAME9/2HDA,2HTA,2H1 /
0175      CALL OPEN(IDCB,IERR,NAME9)
0176      NOGRID=0
0177      IF(NUMC.LE.0) GO TO 1
0178      IF (IALPHA)2,1,3
0179      2 CONTINUE
0180      NOGRID=1
0181      3 CONTINUE
0182      IMAX=IABS(IALPHA)
0183      IMAXP2=IMAX+2
0184      IF (JBETA)4,1,5
0185      4 CONTINUE
0186      NOGRID=1
0187      5 CONTINUE
0188      JMAX=IABS(JBETA)
0189      IF (NOGRID.EQ.1) GO TO 6
0190      DO 7 K=1,4
0191      CALL FLBUF(IXYXYB(1,K),IXYXYB(2,K),
0192      1 IXYXYB(3,K),IXYXYB(4,K),BUFF)
0193      CALL WRITF(IDCB,IERR,BUFF,4)
0194      7 CONTINUE
0195      6 CONTINUE
0196      DO 8 N=1,NUMC
0197      DO 9 I=1,IMAXP2
0198      IXYIY(1,I)=0
0199      JXJY(1,I)=0
0200      9 CONTINUE
0201      DO 10 J=1,JMAX
0202      IF(J.EQ.1) GO TO 111
0203      DO 12 I=1,IMAX
0204      IF(GAMMA(I,J).EQ.GAMMA(I,J-1)) GO TO 13
0205      IF (GAMMA(I,J).GE.C(N).AND.C(N).GE.GAMMA(I,J-1)) GO TO 14
0206      IF(GAMMA(I,J).LE.C(N).AND.C(N).LE.GAMMA(I,J-1)) GO TO 14
0207      13 CONTINUE
0208      JXJY(1,I+1)=0
0209      GO TO 12
0210      14 CONTINUE
0211      BETINT=BETA(J-1)+(BETA(J)-BETA(J-1))*(C(N)-GAMMA(I,J-1))/(GA
0212      1J)-GAMMA(I,J-1)
0213      ALPINT=ALPHA(I)
0214      IXR=IFIX(XZ+AX*ALPINT+BX*BETINT)

```

```

0215      IYR=IFIX(YZ+AY*ALPINT+BY*BETINT)
0216      IF(JXJY(1,I).EQ.0) GO TO 15
0217          CALL FLBUF(IXR,IYR,JXJY(1,I),JXJY(2,I),BUFF)
0218          CALL WRITF(IDC3,IERR,BUFF,4)
0219 15 CONTINUE
0220      IF(IXIY(1,I+1).EQ.0) GO TO 16
0221          CALL FLBUF(IXR,IYR,IXIY(1,I+1),IXIY(2,I+1),BUFF)
0222          CALL WRITF(IDC3,IERR,BUFF,4)
0223 16 CONTINUE
0224      IF(IXIY(1,I+2).EQ.0) GO TO 17
0225          CALL FLBUF(IXR,IYR,IXIY(1,I+2),IXIY(2,I+2),BUFF)
0226          CALL WRITF(IDC3,IERR,BUFF,4)
0227 17 CONTINUE
0228      JXJY(1,I+1)=IXR
0229      JXJY(2,I+1)=IYR
0230 12 CONTINUE
0231 111  CONTINUE
0232      DO 18 I=2,IMAX
0233      IF(GAMMA(I,J).EQ.GAMMA(I-1,J)) GO TO 19
0234      IF (GAMMA(I,J).GE.C(N).AND.C(N).GE.GAMMA(I-1,J)) GO TO 20
0235      IF (GAMMA(I,J).LE.C(N) .AND. C(N).LE.GAMMA(I-1,J)) GO TO 20
0236 19 CONTINUE
0237      IIXIY(1,I+1)=0
0238      GO TO 18
0239 20 CONTINUE
0240      ALPINT=ALPHA(I-1)+(ALPHA(I)-ALPHA(I-1))*(C(N)-GAMMA(I-1,J))/I(I,J)-GAMMA(I-1,J))
0241      BETINT=BETA(J)
0242      IXR=IFIX(XZ+AX*ALPINT+BX*BETINT)
0243      IYR=IFIX(YZ+AY*ALPINT+BY*BETINT)
0244      IF(JXJY(1,I).EQ.0) GO TO 21
0245      CALL FLBUF(IXR,IYR,JXJY(1,I),JXJY(2,I),BUFF)
0246      CALL WRITF(IDC3,IERR,BUFF,4)
0247 21 CONTINUE
0248      IF (IXIY(1,I+1).EQ.0) GO TO 22
0249          CALL FLBUF(IXR,IYR,IXIY(1,I+1),IXIY(2,I+1),BUFF)
0250          CALL WRITF(IDC3,IERR,BUFF,4)
0251 22 CONTINUE
0252      IF(JXJY(1,I+1).EQ.0) GO TO 23
0253          CALL FLBUF(IXR,IYR,JXJY(1,I+1),JXJY(2,I+1),BUFF)
0254          CALL WRITF(IDC3,IERR,BUFF,4)
0255 23 CONTINUE
0256      IIXIY(1,I+1)=IXR
0257      IIXIY(2,I+1)=IYR
0258 18 CONTINUE
0259 10 CONTINUE
0260 8 CONTINUE
0261 1 CONTINUE
0262      RETURN
0263
0264      END

```

```
0265 C ****
0266 C
0267 C
0268 C
0269      SUBROUTINE SET3D(ALPMAX,ALPMIN,BETMAX,BETMIN,GAMMAX,GAMMIN,
0270      ORGN,IALPCL,GAMFAC,ALPFAC)
0271      COMMON/QDCAZ /IXXYXB(4,5),XZ,AX,BX,YZ,AY,BY,CY
0272      DATA ELX,ELY,EXLEFT,YBOTOM/1012.,856.,12.,156./
0273      AX=ALPFAC*ELX/(ALPMAX-ALPMIN)
0274      AY=ALPFAC*(1.-GAMFAC)*ELX/(ALPMAX-ALPMIN)
0275      BX=(1.-ALPFAC)*ELX/(BETMAX-BETMIN)
0276      BY=(1.-ALPFAC)*(1.-GAMFAC)*ELY/(BETMAX-RETMIN)
0277      CY=GAMFAC*ELY/(GAMMAX-GAMMIN)
0278      YZ=-CY*GAMMIN+YBOTOM
0279      XZ=EXLEFT
0280      IF(IORGN.EQ.1)GO TO 1
0281      IF(IORGN.EQ.2)GO TO 2
0282      IF(IORGN.EQ.3)GO TO 3
0283      GO TO 4
0284      1 CONTINUE
0285      XZ=XZ+ELX
0286      AX=-AX
0287      BX=-BX
0288      IF(IALPCL.EQ.1)GO TO 10
0289      YZ=YZ+BY*(BETMAX-BETMIN)
0290      BY=-BY
0291      ALPVRT=ALPMAX
0292      BETVRT=BETMIN
0293      GO TO 5
0294      10 CONTINUE
0295      YZ=YZ+AY*(ALPMAX-ALPMIN)
0296      AY=-AY
0297      ALPVRT=ALPMIN
0298      BETVRT=BETMAX
0299      GO TO 5
0300      2 CONTINUE
0301      ALPVRT=ALPMAX
0302      BETVRT=BETMAX
0303      IF(IALPCL.EQ.1)GO TO 20
0304      XZ=XZ+BX*(BETMAX-BETMIN)
0305      BX=-BX
0306      GO TO 5
0307      20 CONTINUE
0308      XZ=XZ+AX*(ALPMAX-ALPMIN)
0309      AX=-AX
0310      GO TO 5
0311      3 CONTINUE
```

```

0312      IF(IALPCL.EQ.1)GO TO 30
0313      YZ=YZ+AY*(ALPMAX-ALPMIN)
0314      AY=-AY
0315      ALPVRT=ALPMIN
0316      BETVRT=BETMAX
0317      GO TO 5
0318 30  CONTINUE
0319      YZ=YZ+BY*(BETMAX-BETMIN)
0320      BY=-BY
0321      ALPVRT=ALPMAX
0322      BETVRT=BETMIN
0323      GO TO 5
0324 4  CONTINUE
0325      YZ=YZ+ELY*(1.-GAMFAC)
0326      ALPVRT=ALPMIN
0327      BETVRT=BETMIN
0328      AY=-AY
0329      BY=-BY
0330      IF(IALPCL.EQ.1)GO TO 40
0331      XZ=XZ+AX*(ALPMAX-ALPMIN)
0332      AX=-AX
0333      GO TO 5
0334 40  CONTINUE
0335      XZ=XZ+BX*(BETMAX-BETMIN)
0336      BX=-BX
0337 5  CONTINUE
0338      XZ=XZ-BX*BETMIN-AX*ALPMIN
0339      YZ=YZ-BY*BETMIN-AY*ALPMIN
0340      IXYXYB(1,1)=XZ+AX*ALPMIN+BX*BETMIN
0341      IXYXYB(2,1)=YZ+AY*ALPMIN+BY*BETMIN+CY*GAMMIN
0342      IXYXYB(3,1)=XZ+AX*ALPMAX+BX*BETMIN
0343      IXYXYB(4,1)=YZ+AY*ALPMAX+BY*BETMIN+CY*GAMMIN
0344      IXYXYB(1,2)=IXYXYB(3,1)
0345      IXYXYB(2,2)=IXYXYB(4,1)
0346      IXYXYB(3,2)=XZ+AX*ALPMAX+BX*BETMAX
0347      IXYXYB(4,2)=YZ+AY*ALPMAX+BY*BETMAX+CY*GAMMIN
0348      IXYXYB(1,3)=IXYXYB(3,2)
0349      IXYXYB(2,3)=IXYXYB(4,2)
0350      IXYXYB(3,3)=XZ+AX*ALPMIN+BX*BETMAX
0351      IXYXYB(4,3)=YZ+AY*ALPMIN+BY*BETMAX+CY*GAMMIN
0352      IXYXYB(1,4)=IXYXYB(3,3)
0353      IXYXYB(2,4)=IXYXYB(4,3)
0354      IXYXYB(3,4)=IXYXYB(1,1)
0355      IXYXYB(4,4)=IXYXYB(2,1)
0356      IXYXYB(1,5)=XZ+AX*ALPVRT+BX*BETVRT
0357      IXYXYB(2,5)=YZ+AY*ALPVRT+BY*BETVRT+CY*GAMMIN
0358      IXYXYB(3,5)=IXYXYB(1,5)
0359      IXYXYB(4,5)=YZ+AY*ALPVRT+BY*BETVRT+CY*GAMMAX
0360 45      FORMAT (5(7X,I7))
0361      RETURN
0362      END

```

```

0363 C ****
0364 C
0365 C
0366 C
0367 SUBROUTINE PLT3D(ALPHA,BETA,GAMMA,IMDN,IALPHA,JBETA,IFILE,L
0368 DIMENSION ALPHA(1),BETA(1),GAMMA(IDMN,1)
0369 COMMON/WORK/LASTXY(2,200)
0370 COMMON/QDCAL /IXYXYB(4,5),XZ,AX,BX,YZ,AY,BY,CY
0371 INTEGER BUFF(4)
0372 COMMON IDCBL(144)
0373 NOGRID=0
0374 IF(IALPHA)1,2,3
0375 1 CONTINUE
0376 NOGRID=1
0377 3 CONTINUE
0378 IMAX=IABS(IALPHA)
0379 IF(JBETA)4,2,5
0380 4 CONTINUE
0381 NOGRID=1
0382 5 CONTINUE
0383 JMAX=IABS(JBETA)
0384 IF(NOGRID.EQ.1)GO TO 6
0385 67 FORMAT (5(7X,I7))
0386 68 FORMAT (10X,"GOOD",I5)
0387 DO 7 K=1,5
0388 CALL FLBUF( IXYXYB(1,K),IXYXYB(2,K),
0389 1 IXYXYB(3,K),IXYXYB(4,K),BUFF)
0390 CALL WRITF(IDCBL,IERR,BUFF,4)
0391 7 CONTINUE
0392 6 CONTINUE
0393 DO 8 J=1,JMAX
0394 DO 8 I=1,IMAX
0395 IXR=IFIX(XZ+AX*ALPHA(I)+BX*BETA(J))
0396 IYR=IFIX(YZ+AY*ALPHA(I)+BY*BETA(J)+CY*GAMMA(I,J))
0397 IF(I.EQ.1)GO TO 9
0398 CALL FLBUF(IXR,IYR,LASTXY(1,I-1),LASTXY(2,I-1),BUFF)
0399 CALL WRITF(IDCBL,IERR,BUFF,4)
0400 9 CONTINUE
0401 IF(J.EQ.1)GO TO 10
0402 CALL FLBUF(IXR,IYR,LASTXY(1,I),LASTXY(2,I),BUFF)
0403 CALL WRITF(IDCBL,IERR,BUFF,4)
0404 10 CONTINUE
0405 LASTXY(1,I)=IXR
0406 LASTXY(2,I)=IYR
0407 8 CONTINUE
0408 2 CONTINUE
0409 RETURN
0410 END
0411 SUBROUTINE FLBUF(IX1,IY1,IX2,IY2 ,BUFF)
0412 INTEGER BUFF(4)
0413 BUFF(1) = IX1
0414 BUFF(2) = IY1
0415 BUFF(3) = IX2
0416 BUFF(4) = IY2
0417 RETURN
0418 END
0419 $

```

&FDIGN T=00004 IS ON CR00022 USING 00056 BLKS R=0498

```

0001  FTN4,L
0002      PROGRAM FDIGN
0003  C
0004  C THIS PROGRAM SCHEDULE FILTER DESIGN,STABILITY,AND DISPLAY
0005  C
0006      COMMON/WORK/W0(75)
0007      DIMENSION NAME1(3),NAME2(3),NAME3(3),IRTN(5)
0008      DIMENSION IDC8(144),NAME4(3),NAME5(3)
0009      DIMENSION U(3,3,2),V(3,3,2),ILU(5),IBUF(80),IBUF2(80)
0010      EQUIVALENCE (ILU(1),LU),(IBUF,IBUF2)
0011      EQUIVALENCE (IBUF(1),U(1,1,1)),(IBUF(41),V(1,1,1))
0012      DATA NAME1/2HST,2HAB,2HI /
0013      DATA NAME2/2HDP,2HLA,2HM /
0014      DATA NAME3/2HCO,2HEF,2HFS/
0015      DATA NAME4/2HFI,2HRO,2H /
0016      DATA NAME5/2HPL,2HOT,2HV /
0017      DATA V/18*0./
0018      DATA U/18*0./
0019      DATA IBUF/80*0/
0020  C
0021  C GET LU
0022      CALL RMPAR(ILU)
0023  C
0024  C      GET FILTER PARAMETERS
0025  C
0026      MN=1
0027  4      WRITE(LU,400)
0028  400 FORMAT(" SELECT FILTER DESIGN"/" 1. LOWPASS"/" 2. BANDPA
0029      1 3. HIGHPASS"/" 4. BOOST FILTER"/" 5. TDLPF (LOWPASS)"
0030      2" 6. ROTATING FILTER "/" 7. NON-RECURSIVE FILTERS ")
0031      READ(LU,401) IFIL
0032      IF(IFIL.EQ.4) GO TO 500
0033      IF(IFIL.EQ.3)GO TO 410
0034          IF(IFIL.EQ.6) GO TO 408
0035          IF(IFIL .EQ. 7) GO TO 1102
0036      WRITE(LU,402)
0037  402 FORMAT(" ENTER RELATIVE CUTOFF FREQUENCY FOR LOWPASS")
0038      READ(LU,403)F2
0039  403 FORMAT(F2.2)
0040      IF(IFIL.NE.2) GO TO 407
0041      WRITE(LU,404)
0042  404 FORMAT(" ENTER RELATIVE CUTOFF FREQUENCY FOR HIGHPASS")
0043      READ(LU,403) F1
0044  407 WRITE(LU,405)
0045  405 FORMAT(" ENTER NUMBER OF FILTER STAGES")
0046      READ(LU,401) MN
0047      IBUF(40) =MN
0048  401 FORMAT(1I1)
0049      GO TO 411
0050  500 WRITE(LU,510)
0051  510 FORMAT(" SELECT OPTION"/," 1. LOW BOOST FILTER"/," 2. HI
0052      *ST FILTER")
0053      PEAD(LU,515) IOPT
0054  515 FORMAT(1I1)
0055      IF(IOPT.GE.0.AND.IOPT.LE.2) GO TO 530

```

```

0056      WRITE(LU,520)
0057      520 FORMAT(" INVALID RESPONSE")
0058      GO TO 500
0059  C
0060      530 WRITE(LU,535)
0061      535 FORMAT(" ENTER BOOST MAGNITUDE")
0062      READ(LU,*) BF
0063      WRITE(LU,540)
0064      540 FORMAT(" ENTER RELATIVE BREAK FREQUENCY")
0065      READ(LU,403) F1
0066      IF(IOPT.EQ.0.OR.IOPT.EQ.1) WRITE(LU,545) BF,F1
0067      545 FORMAT(" BOOST MAGNITUDE = ",1E15.5," FREQUENCY = ",1F10.5,
0068      *OWBOOST FILTER."/," IS THIS CORRECT?")
0069      IF(IOPT.EQ.2) WRITE(LU,550) BF,F1
0070      550 FORMAT(" BOOST MAGNITUDE = ",1E15.5," BREAK FREQUENCY = ",1F
0071      * FOR HIGH BOOST FILTER."/," IS THIS CORRECT?")
0072      READ(LU,551) IRES
0073      551 FORMAT(1A1)
0074      IF(IRES.EQ.1HY.OR.IRES.EQ.1Hy) GO TO 552
0075      GO TO 530
0076      552 BF=SQRT(ABS(BF))
0077      IF(IOPT.EQ.2) GO TO 560
0078      ALP=1.0
0079      BET=BF-1.0
0080      GO TO 412
0081      560 ALP=BF
0082      BET=1.0-BF
0083      GO TO 412
0084      410 WRITE(LU,404)
0085      READ(LU,403) F1
0086      412 WRITE(LU,405)
0087      READ(LU,401) MN
0088      411 MN=MN+1
0089      408 IF(IFIL.EQ.1) CALL LPFLT(U,V,F2,MN,LU)
0090      IF(IFIL.EQ.2) CALL BPFLT(U,V,F1,F2,MN,LU)
0091      IF(IFIL.EQ.3) CALL BSTFT(U,V,F1,MN,1.0,-1.0,LU)
0092      IF(IFIL.EQ.4) CALL BSTFT(U,V,F1,MN,ALP,BET,LU)
0093      IF(IFIL.EQ.5) CALL TDLPF(U,V,MN,F2,2,LU)
0094  IL.E   CALL BPFLT(U,V,F1,F2,MN,LU)
0095  IL.E   IF(IFIL.EQ.3) CALL BSTFT(U,V,F1,MN,1.0,-1.0,LU)
0096  IL.E   IF(IFIL.EQ.4) CALL BSTFT(U,V,F1,MN,ALP,BET,LU)
0097  IL.E   IF(IFIL.EQ.5) CALL TDLPF(U,V,MN,F2,2,LU)
0098  IL.E   IF(IFIL.EQ.6) CALL ROTAE(U,V,MN,LU)
0099  C     IF(IFIL.EQ.7) CALL FIR(U,MN,WR,LU)
0100  C     CONTINUE
0101  C     ILU(2) = MN
0102  C     IF(IFIL.EQ.2) ILU(2) =MN + 1
0103  C     IF(IFIL.EQ.6) ILU(2)=MN-1
0104  C
0105  C
0106  C     SCHEDULE DISPLAY PROGRAM-DPLAY
0107  C     IF(IFIL.EQ.3) ILU(2) =MN + 1
0108  C     IF(IFIL.EQ.4) ILU(2) =MN+1
0109  C     CALL EXEC(23,NAME2,LU,ILU(2),0,0,0,IBUF2,80)
0110  C

```

```

0111      IF(IFIL .EQ. 3) IBUF(40) =MN
0112      IF(IFIL .EQ. 4) IBUF(40) =MN
0113      IF(IFIL .EQ. 6) IBUF(40)=MN-1
0114      C
0115      CALL PURGE(IDCDB,IERR,NAME3,2HES)
0116      IF(IERR .LT. 0) WRITE(LU,1101) IERR
0117      CALL CREAT(IDCDB,IERR,NAME3,2,3,2HES)
0118      IF(IERR .LT. 0) WRITE(LU,1101) IERR
0119      1101  FORMAT("CREATE ERROR",F5.0)
0120      CALL WRITF(IDCDB,IERR,IBUF,80)
0121      IF(IERR .LT. 0) WRITE(LU,1101) IERR
0122      CALL CLOSE(IDCDB,IERR)
0123      WRITE(LU,1105)
0124      1105  FORMAT(" ENTER DISPLAY DEVICE //" 1. TV"/" 2. HP2648A")
0125      READ(LU,*) IDEV
0126      C
0127      IF(IDEV .EQ. 2) GO TO 1106
0128      CALL EXEC(23,NAME5,LU,0,0,0,0)
0129      GO TO 1107
0130      1106  CONTINUE
0131      CALL HP48A(LU)
0132      1107  CONTINUE
0133      CALL EXEC(6)
0134      C
0135      C SCHEDULE NON-RECURSIVE FILTERS
0136      1102  CONTINUE
0137      CALL EXEC(23,NAME4,LU,0,0,0,0)
0138      STOP
0139      END
0140      SUBROUTINE BPFLT(U,V,F1,F2,N,LU)
0141      C
0142      C          WRITTEN BY W. E. ALEXANDER
0143      C
0144      C          F1----BREAK FREQUENCY FOR LOW FREQUENCY CUTOFF
0145      C          F2----BREAK FREQUENCY FOR HIGH FREQUENCY CUTOFF
0146      C
0147      C          SUBROUTINE DESIGNS BANDPASS FILTER FROM LPFLT AND HPFLT
0148      C
0149      DIMENSION U(3,3,2),V(3,3,2),AA(3,3,2),BB(3,3,2)
0150      IF(F1.LT.0.001.OR.F1.GT.0.999) RETURN
0151      IF(F2.LT.0.001.OR.F2.GT.0.999) RETURN
0152      C
0153      FC=AMAX1(F1,F2)
0154      CALL LPFLT(AA,BB,FC,N,LU)
0155      C
0156      DO 20 I=1,3
0157      DO 20 J=1,3
0158      U(I,J,1)=AA(I,J,1)
0159      20 V(I,J,1)=BB(I,J,1)
0160      C
0161      FC=AMIN1(F1,F2)
0162      CALL BSTFT(AA,BB,FC,N,1.0,-1.0,LU)

```

```

0163 C
0164      DO 21 I=1,3
0165      DO 21 J=1,3
0166      U(I,J,2)=AA(I,J,1)
0167      21 V(I,J,2)=BB(I,J,1)
0168 C
0169
0170      RETURN
0171      END
0172      SUBROUTINE BSTFT(U,V,FC,N,ALP,BET,LU)
0173 C      FREQUENCY BOOST DESIGN ROUTINE
0174 C      FC=RC*S/PI WHERE RC IS THE CUTOFF FREQUENCY IN RADIANS AND
0175 C      S IS THE SAMPLING INTERVAL.  THUS FC=0.5 GIVES A CUTO
0176 C      FREQUENCY AT ONE FOURTH SAMPLING FREQUENCY.
0177 C
0178 C      FOR HIGH PASS FILTER, LET ALP=1.0 AND BET=-1.0
0179 C      FOR HIGH FREQUENCY BOOST FILTER, ALP=BF AND BET=-1.0*(BF-1.0
0180 C          WHERE BF=SQRT(DESIRED FILTER GAIN AT ONE HALF SAMPLING
0181 C      FOR LOW FREQUENCY BOOST FILTER, ALP=1.0 AND BET=(BF-1.0)
0182 C
0183 C
0184      DIMENSION U(3,3,2),V(3,3,2)
0185 C
0186      DO 21 K=1,2
0187      DO 21 I=1,3
0188      DO 21 J=1,3
0189      U(I,J,K)=0.0
0190      21 V(I,J,K)=0.0
0191      WRITE(LU,14) FC
0192      14 FORMAT(1H0," FC = ",1E22.5," FOR BOOST FILTER"/)
0193      PI=3.141592654
0194      D=1.0E-10
0195      PWR=0.25
0196      IF (N.EQ.3) PWR=0.125
0197      EPS=2.0**PWR-1.0
0198      IF(N.EQ.2.AND.BET.LT.0.0) EPS=1.50702
0199      IF(N.EQ.3.AND.BET.LT.0.0) EPS=2.4711
0200      XP=PI*FC*0.5
0201      T=(SIN(XP)/COS(XP))**2.0
0202      IF(T.GT.D) GO TO 10
0203      AAA=EPS/D
0204      GO TO 11
0205      10 AAA=EPS/T
0206      11 SALP=SQRT(AAA)
0207      DEM=1.0-2.0*AAA
0208      IF(DEM.LT.D) GO TO 12
0209      13 P1=(+2.0*AAA-2.0*SQRT(2.0)*SALP+1.0)/DEM
0210      GO TO 20
0211      12 F=-1.0*DEM
0212      IF(F.GT.D) GO TO 13
0213      P1=0.0
0214 C
0215      20 A=((1.0+P1)**2)/4.0
0216      AS=A**2
0217      POS=P1**2
0218      R=4.0*(POS-AS)+((1.0+POS)**2-4.0*AS)-4.0*(P1*(1.0+POS)-2.0*A
0219      IF(ABS(R).LT.D) R=SIGN(D,R)
0220      S=((1.0-P1)**4)/R
0221 C

```

```

0222      U(1,1,1)=S*(ALP*POS+BET*AS)
0223      U(1,2,1)=S*(ALP*P1*(1.0+POS)+2.0*BET*AS)
0224      U(2,1,1)=U(1,2,1)
0225      U(2,2,1)=S*(ALP*(1.0+POS)**2+4.0*BET*AS)
0226      U(1,3,1)=S*(ALP*POS+BET*AS)
0227      U(3,1,1)=U(1,3,1)
0228      U(2,3,1)=S*(ALP*P1*(1.0+POS)+2.0*BET*AS)
0229      U(3,2,1)=U(2,3,1)
0230      U(3,3,1)=S*(ALP*POS+BET*AS)
0231 C
0232      V(1,1,1)=1.0
0233      V(1,2,1)=2.0*P1
0234      V(2,1,1)=V(1,2,1)
0235      V(2,2,1)=4.0*POS
0236      V(1,3,1)=POS
0237      V(3,1,1)=V(1,3,1)
0238      V(2,3,1)=2.0*P1*POS
0239      V(3,2,1)=V(2,3,1)
0240      V(3,3,1)=POS**2
0241 C
0242      IF(N.EQ.2) GO TO 27
0243      DO 26 I=1,3
0244      DO 26 J=1,3
0245      U(I,J,2)=U(I,J,1)
0246      26 V(I,J,2)=V(I,J,1)
0247      27 N = N-1
0248      RETURN
0249      END
0250      SUBROUTINE LPFLT(U,V,FC,N,LU)
0251 C      LOW PASS RECURSIVE FILTER DESIGN ROUTINE
0252 C      FC/RC*S/PI WHERE RC IS THE CUTOFF FREQUENCY IN RADIANS AND
0253 C      S IS THE SAMPLING INTERVAL.  THUS FC#0.5 GIVES A CUTO
0254 C      FREQUENCY AT ONE FOURTH SAMPLING FREQUENCY.
0255 C
0256      DIMENSION U(3,3,2),V(3,3,2)
0257      COMMON/WORK/A(5,5),B(5,5)
0258 C
0259      DO 21 K=1,2
0260      DO 21 I=1,3
0261      DO 21 J=1,3
0262      U(I,J,K)=0.0
0263      21 V(I,J,K)=0.0
0264      IF(FC.GE.0.99) FC=0.99
0265      WRITE(LU,14)FC
0266      14 FORMAT(1HO," FC = ",1E10.4," FOR LOWPASS FILTER ",/)
0267      PI=3.141592654
0268      D=1.0E-10
0269      PWR=0.25
0270      IF (N.EQ.3) PWR=0.125
0271      EPS=2.0**PWR-1.0
0272      XP=PI*FC*0.5
0273      T=(SIN(XP)/COS(XP))**2.0
0274      IF(T.GT.D) GO TO 10
0275      ALP=EPS/D
0276      GO TO 11
0277      10 ALP=EPS/T
0278      11 SALP=SQRT(ALP)
0279      DEM=1.0-2.0*ALP
0280      IF(DEM.LT.D) GO TO 12
0281      13 P1=(+2.0*ALP-2.0*SQRT(2.0)*SALP+1.0)/DEM

```

```

0282      P2=P1
0283      IF(FC.GT.0.3)GO TO 20
0284      P2=(SQRT(T)-EPS)/(SQRT(T)+EPS)
0285      GO TO 20
0286 12  T=-1.0*DEM
0287      IF(F.GT.D) GO TO 13
0288      P1=0.0
0289      P2=0.0
0290 20  V(1,1,1)=1.0
0291      V(1,2,1)=P1+P2
0292      V(2,1,1)=V(1,2,1)
0293      V(1,3,1)=P1*P2
0294      V(3,1,1)=V(1,3,1)
0295      V(2,2,1)=V(1,2,1)**2
0296      V(2,3,1)=P1*P2**2+P2*P1**2
0297      V(3,2,1)=V(2,3,1)
0298      V(3,3,1)=V(1,3,1)**2
0299  C
0300      SUM=0.0
0301      DO 25 I=1,3
0302      DO 25 J=1,3
0303 25  SUM=SUM+V(I,J,1)
0304  C
0305      U(1,1,1)=SUM/16.0
0306      U(1,3,1)=U(1,1,1)
0307      U(3,1,1)=U(1,1,1)
0308      U(3,3,1)=U(1,1,1)
0309      U(1,2,1)=SUM/8.0
0310      U(2,1,1)=U(1,2,1)
0311      U(2,3,1)=U(1,2,1)
0312      U(3,2,1)=U(1,2,1)
0313      U(2,2,1)=SUM/4.0
0314      IF(N.EQ.2) GO TO 1
0315      DO 26 I=1,3
0316      DO 26 J=1,3
0317      U(I,J,2)=U(I,J,1)
0318 26  V(I,J,2)=V(I,J,1)
0319      GO TO 2
0320      1 U(1,1,2)=1.0
0321      V(1,1,2)=1.0
0322      2 DO 30 I=1,5
0323      DO 30 J=1,5
0324      A(I,J)=0.0
0325 30  B(I,J)=0.0
0326      DO 31 I=1,5
0327      DO 31 J=1,5
0328      DO 31 K=1,3
0329      DO 31 L=1,3
0330      IK=I-K+1
0331      JL=J-L+1
0332      IF(IK.LE.0.OR.IK.GT.3)GO TO 31
0333      IF(JL.LE.0.OR.JL.GT.3)GO TO 31
0334      A(I,J)=A(I,J)+U(IK,JL,1)*U(K,L,2)
0335      B(I,J)=B(I,J)+V(IK,JL,1)*V(K,L,2)
0336 31  CONTINUE
0337      RETURN
0338      END

```

```

0339      SUBROUTINE TDLPF(A,B,MN,RC,NDIM,LU)
0340      C      INPUTS
0341      C          N - NUMBER OF FILTER STAGES
0342      C          RC - RELATIVE CUTOFF FREQUENCY FOR FILTER
0343      C          NDIM - ARRAY DIMENSION IN CALLING PROGRAM
0344      C      OUTPUTS
0345      C          A - COEFFICIENT ARRAY (NUMERATOR)
0346      C          B - COEFFICIENT ARRAY (DENOMINATOR)
0347      C
0348      C          DIMENSION A(3,3,NDIM),B(3,3,NDIM)
0349      C          COMPLEX P(10),PK,Q,Z1,Z2
0350      C
0351      C      INITIALIZE
0352      C
0353      N=MN-1
0354      PI=3.141592654
0355      D=1.0E-10
0356      IF(N.GT.NDIM) GO TO 300
0357      IF(0.01.GT.RC.OR.0.99.LT.RC) GO TO 400
0358      AA=0.5*PI*RC
0359      AA=SIN(AA)/COS(AA)
0360      PWR=1.0/FLOAT(N)
0361      C      EPS=SQRT(2.0)-1.0
0362      EPS=1.0
0363      EPS=EPS**PWR
0364      C=AA**2/EPS
0365      C
0366      C      FIND ROOTS
0367      C
0368      L=1
0369      NN=2.0*N
0370      CONST=FLOAT(NN+1)/FLOAT(NN)
0371      DO 10 K=1,NN
0372      THETA=PI*(1.0+2.0*(K-1))*CONST
0373      PK=C*CMPLX(COS(THETA),SIN(THETA))
0374      C      WRITE(LU,14) PK
0375      14 FORMAT(" PK = ",1E15.5," + J",1E15.5/)
0376      Q=2.0-PK
0377      IF(CABS(Q).LE.D) Q=D
0378      IF(CABS(PK).LE.D) PK=SIGN(D,REAL(PK))
0379      Z1=(2.0+PK+2.0*CSQRT(2.0*PK))/Q
0380      C      WRITE(LU,12) Z1
0381      12 FORMAT(" Z1 = ",1E15.5," + J",1E15.5/)
0382      IF(CABS(Z1).GE.1.0) GO TO 15
0383      P(L)=Z1
0384      C      WRITE(LU,11) L,P(L)
0385      11 FORMAT(" P(",1I2,") = ",1E15.5," + J",1E15.5/)
0386      L=L+1
0387      15 Z2=(2.0+PK-2.0*CSQRT(2.0*PK))/Q
0388      C      WRITE(LU,13) Z2
0389      13 FORMAT(" Z2 = ",1E15.5," + J",1E15.5/)
0390      IF(CABS(Z2).GE.1.0) GO TO 20
0391      P(L)=Z2
0392      C      WRITE(LU,11) L,P(L)
0393      L=L+1
0394      20 IF((L-1).EQ.NN) GO TO 25
0395      10 CONTINUE

```

```

0396 C
0397 C      PAIR COMPLEX PAIRS OF ROOTS
0398 C
0399 25 L=1
0400      DO 30 K=1,NN
0401      S1=AIMAG(P(K))
0402      IF(S1.LT.0.0) GO TO 30
0403      P(L)=P(K)
0404      L=L+1
0405 30 CONTINUE
0406 C
0407 C      OBTAIN FILTER COEFFICIENTS
0408 C
0409      IF((L-1).LT.N) GO TO 500
0410      DO 40 K=1,N
0411      C1=-2.0*REAL(P(K))
0412      C2=CABS(P(K))**2
0413      AM=(1.0+C1+C2)**2/16.0
0414 C
0415      A(1,1,K)=AM
0416      A(1,2,K)=2.0*AM
0417      A(2,1,K)=2.0*AM
0418      A(1,3,K)=AM
0419      A(3,1,K)=AM
0420      A(2,2,K)=4.0*AM
0421      A(2,3,K)=2.0*AM
0422      A(3,2,K)=2.0*AM
0423      A(3,3,K)=AM
0424 C
0425      B(1,1,K)=1.0
0426      B(2,1,K)=C1
0427      B(1,2,K)=C1
0428      B(1,3,K)=C2
0429      B(3,1,K)=C2
0430      B(2,2,K)=C1**2
0431      B(2,3,K)=C1*C2
0432      B(3,2,K)=C1*C2
0433 40 B(3,3,K)=C2**2
0434 C
0435      GO TO 600
0436 300 WRITE(LU,310)
0437 310 FORMAT(" NUMBER OF STAGES TOO LARGE FOR DIMENSION")
0438      GO TO 600
0439 400 WRITE(LU,410)
0440 410 FORMAT(" FREQUENCY SPECIFICATION OUT OF RANGE")
0441      GO TO 600
0442 500 WRITE(LU,510)
0443 510 FORMAT(" NUMBER OF ROOTS LESS THAN EXPECTED")
0444      600 RETURN
0445      END

```

```
0446      BLOCK DATA WORK
0447      COMMON/WORK/W0(75)
0448      END
0449      SUBROUTINE HP48A(J)
0450      DIMENSION IB(14),IA(4)
0451      INTEGER IDCB(144),BUFF( 4 ), NAME(3)
0452      DATA NAME/2HDA,2HTA,2H1 /
0453      C
0454      CALL OPEN(IDCB,IERR,NAME)
0455      IF (IERR .GE. 0) GO TO 30
0456      WRITE(LU,10) IERR
0457      10  FORMAT ("OPEN ERROR",F5.0)
0458      STOP
0459      30  CALL GRAFC(1,LU)
0460      20  CALL READF(IDCB,IERR,BUFF,4,ILOG)
0461      IF(ILOG .EQ. -1) GO TO 55
0462      IF (IERR .GE. 0) GOTO 40
0463      WRITE(LU,31) IERR
0464      31  FORMAT("READ ERROR",F5.0)
0465      GO TO 55
0466      40  CONTINUE
0467      CALL DVECT(BUFF,BUFF(2),BUFF(3),BUFF(4),LU)
0468      50  GO TO 20
0469      55  CALL EXEC(13,LU,ISTAT)
0470      ISTAT=IAND(ISTAT,140000B)
0471      IF(ISTAT.NE.0) GO TO 55
0472      CALL GRAFC(0,LU)
0473      CALL CLOSE(IDCB)
0474      RETURN
0475      END
0476      SUBROUTINE GRAFC(IFLAG,LU)
0477      INTEGER IESC
0478      IESC= 33B
0479      C
0480      C GRAPHICS OFF=0; GRAPHICS ON NOT=0
0481      C
0482      IF(IFLAG.EQ.0) GO TO 100
0483      C
0484      C GRAPHIC ON
0485      C
0486      WRITE(LU,10) IESC
0487      10  FORMAT(1R2,"*dC")
0488      WRITE(LU,12) IESC
0489      12  FORMAT(1R2,"*dF")
0490      WRITE(LU,14) IESC
0491      14  FORMAT(1R2,"*dA")
0492      C
```

```
0493      GO TO 200
0494  C
0495  C  GRAPHICS OFF
0496  C
0497 100      WRITE(LU,30) IESC
0498 30      FORMAT(1R2,"*d")
0499      WRITE(LU,40) IESC
0500 40      FORMAT(1R2,"*dE")
0501 200      RETURN
0502      END
0503      SUBROUTINE DVECT(IX1,IY1,IX2,IY2,LU)
0504  C
0505  C      SUBROUTINE DRAWS A LINE BETWEEN THE TWO POINTS (IX1,IY1)
0506  C          AND (IX2,IY2).  THE POINT (IX0,IY0) DEFINES THE
0507  C          THE ORIGIN.
0508  C
0509      IX0=0
0510      IY0=0
0511      XSCAL =356.0/1024.0
0512      YSCAL =XSCAL
0513      X1 = IX1*XSCAL + 0.5
0514      X2 = IX2*XSCAL + 0.5
0515      Y1 = IY1*YSCAL + 0.5
0516      Y2 = IY2*YSCAL + 0.5
0517      JX1 = X1 + IX0
0518      JX2 = X2 + IX0
0519      JY1 = Y1 + IY0
0520      JY2 = Y2 + IY0
0521      WRITE(LU,10) JX1,JY1,JX2,JY2
0522 10      FORMAT("pa",1I3,1H,,1I3,1H,,1I3,1I,,1I3,"z")
0523      RETURN
0524      END
0525      END$
0526  $
```

&BLDIM T=00004 IS ON CR00022 USING 00034 BLKS R=0330

```
0001  FTN4
0002      PROGRAM BLDIM
0003  C
0004  C  THIS PROGRAM BUILDS AN IMAGE FILE FOR THE NCA&T IMAGE DISPLAY
0005  C  SYSTEM.  IMAGE FILES MAY BE GENERATED FROM THE GMR-27 DISPLAY,
0006  C  TAPES OR DISC (TYPE 2 FILES).
0007  C
0008  C  PROGRAMMER: DLJ
0009  C
0010      DIMENSION LU(5),IDCB1(272),IDCB2(528),NAME(6),ISIZE(2),IDATA
0011      DIMENSION JNAME(3),IBUF(6)
0012  C
0013      INTEGER ENTRY(256),TEXT1(40),TEXT2(40),TEXT3(40),RDREC
0014  C
0015      EQUIVALENCE (ENTRY,NAME),(ENTRY(7),NLINE),(ENTRY(8),NPIXL),
0016      1 (ENTRY(9),IPMIN),(ENTRY(10),IPMAX),(ENTRY(11),ISRC),
0017      2 (ENTRY(13),JNAME),(ENTRY(129),TEXT1),(ENTRY(169),TEXT2),
0018      3 (ENTRY(209),TEXT3),(ENTRY(12),ILOC)
0019      EQUIVALENCE (JNAME(2),JNAM2),(JNAME(3),JNAM3),(ISIZE(2),ISIZ
0020  C
0021  C  CONSTANTS
0022  C      MPIXL = MAXIMUM PIXELS/LINE (WHEN CHANGING BE SURE TO MODIF
0023  C          ARRAY SIZES)
0024  C
0025      DATA MPIXL/512/
0026  C
0027  C  GET INPUT PARAMETERS
0028  C
0029      CALL RMPAR(LU)
0030      IF (LU .LE. 0) LU = 1
0031  C
0032  C  OUTPUT HEADING
0033  C
0034      WRITE(LU,1)
0035  1      FORMAT(//"      B U I L D      I M A G E      S U B S Y S T E M")
0036  C
0037  C  OPEN DIRECTORY FILE
0038  C
0039      CALL OPEN(IDCB1,IERR,6HIMDIRC,0,2HIM,23,272)
0040      IF (IERR .LT. 0) GO TO 9999
0041  C
0042  C  GET IMAGE NAME
0043  C
0044  1000  WRITE(LU,2)
0045  2      FORMAT("ENTER 12 CHARACTER IMAGE NAME?(/E TO EXIT) ")
0046      READ(LU,`` NAME
0047  3      FORMAT(6A2)
0048      IF (NAME .EQ. 2H/E) GO TO 1060
0049  C
0050  C  CHECK FOR DUPLICATE NAME
0051  C
```

```

0052      IREC = 0
0053      KREC = 0
0054      CALL RWNDF(IDCB1,IERR)
0055      IF (IERR .LT. 0) GO TO 9999
0056 1010  IREC = IREC + 1
0057      CALL READF(IDCB1,IERR,IBUF,6,LEN)
0058      IF (IERR .LT. 0) GO TO 9999
0059      IF (LEN .EQ. -1) GO TO 1030
0060  C
0061  C  COMPARE NAME
0062  C
0063      IF (IBUF .EQ. -1) KREC = IREC
0064  C
0065      DO 1020 I=1,6
0066      IF (NAME(I) .NE. IBUF(I)) GO TO 1010
0067 1020  CONTINUE
0068  C
0069  C  DUPLICATE NAME FOUND
0070  C
0071      WRITE(LU,4)
0072  4      FORMAT("ERROR-DUPLICATE NAME")
0073      CALL RWNDF(IDCB1,IERR)
0074      IF (IERR .LT. 0) GO TO 9999
0075      GO TO 1000
0076  C
0077  C  EOF REACHED AND NO DUPLICATE FOUND
0078  C
0079  C  GET IMAGE PARAMETERS
0080  C
0081 1030  WRITE(LU,5)
0082  5      FORMAT("# LINES IN IMAGE?_")
0083      READ(LU,*) NLINE
0084      WRITE(LU,6)
0085  6      FORMAT("# PIXELS/LINE?_")
0086      READ(LU,*) NPIXL
0087      IF (NPIXL .GT. MPIXL) NPIXL = MPIXL
0088  C
0089  C  GET 3-LINES OF DESCRIPTIVE TEXT
0090  C
0091      WRITE(LU,7)
0092  7      FORMAT(" ENTER UP TO 3 LINES OF DESCRIPTIVE TEXT/")
0093      TEXT1 = 2H
0094      CALL MVW(TEXT1,TEXT1(2),119)
0095      CALL EXEC(1,400B+LU,TEXT1,40)
0096      CALL EXEC(1,400B+LU,TEXT2,40)
0097      CALL EXEC(1,400B+LU,TEXT3,40)
0098  C
0099  C  GET SOURCE OF IMAGE
0100  C
0101 1040  WRITE(LU,8)
0102  8      FORMAT("IMAGE SOURCE?(1=DISC FILE;2=TAPE;3=DISPLAY;4=WORK FI
0103      READ(LU,*) ISRC
0104      IF (ISRC .LT. 0) GO TO 1060
0105      IF (ISRC .LT. 1 .OR. ISRC .GT. 4) GO TO 1040
0106  C

```

```
0107 C CREATE DATA FILE
0108 C
0109     ISIZE = (FLOAT(NPIXL)*FLOAT(NLINE) + 127.)/ 128.
0110     ISIZ2 = NPIXL
0111     IF (KREC .EQ. 0) KREC = IREC
0112     JNAME = 2HIM
0113     CALL DCODE(KREC,JNAM2,JNAM3)
0114     CALL PURGE(IDCBL,IERR,JNAME,2HIM,23)
0115     CALL CREAT(IDCBL,IERR,JNAME,ISIZE,2,2HIM,23,528)
0116     IF (IERR .LT. 0) GO TO 9999
0117 C
0118 C INITIALIZE INPUT ROUTINE
0119 C
0120     IERR = RDREC(-LU,ISRC,NLINE,NPIXL)
0121     IF (IERR .LT. 0) GO TO 9999
0122 C
0123 C GET EACH LINE AND WRITE TO FILE
0124 C
0125     IPMAX = 0
0126     IPMIN = 377B
0127     DO 1050 I=1,NLINE
0128     IERR = RDREC(1,IData,IPMAX,IPMIN)
0129     IF (IERR .LT. 0) GO TO 9999
0130     CALL WRITF(IDCBL,IERR,IData,NPIXL)
0131     IF (IERR .LT. 0) GO TO 9999
0132 C     WRITE(LU,1051) IPMAX,IPMIN
0133 1051 FORMAT(2I12)
0134 1050 CONTINUE
0135 C
0136     CALL CLOSE(IDCBL)
0137 C
0138 C WRITE DIRECTORY ENTRY
0139 C
0140     IF (KREC .EQ. IREC) GO TO 1055
0141     CALL OPEN(IDCBL,IERR,6HIMDIRC,2,2HIM,23,272)
0142     IF (IERR .LT. 0) GO TO 9999.
0143     CALL POSNT(IDCBL,IERR,KREC)
0144     IF (IERR .LT. 0) GO TO 9999
0145 1055 ILOC = 1
0146     CALL WRITF(IDCBL,IERR,ENTRY,256)
0147     IF (IERR .LT. 0) GO TO 9999
0148     GO TO 1000
0149 C
0150 C TERMINATE
0151 C
0152 1060 CALL CLOSE(IDCBL)
0153     CALL EXEC(6)
0154 C
0155 C ERROR
0156 C
0157 9999 WRITE(LU,9)IERR
0158 9     FORMAT(" FILE ERROR-",I6)
0159     CALL CLOSE(IDCBL)
0160     END
```

```
0161      INTEGER FUNCTION RDREC(ICODE,IBUF,IP1,IP2)
0162      C
0163      C THIS SUBROUTINE IS USED TO INPUT IMAGE FROM DISC,TAPE OR DISPLA
0164      C
0165      DIMENSION IBUF(1),IDATA(1024),NAME(3),RDATA(512),IDCB(1040)
0166      C
0167      LOGICAL PACKED
0168      C
0169      EQUIVALENCE (IDATA,RDATA)
0170      C
0171      C
0172      IF (ICODE .GT. 0) GO TO 120
0173      C
0174      C INITIALIZATION
0175      C
0176      NLINE = IP1
0177      NPIXL = IP2
0178      LU = -ICODE
0179      C
0180      IF (LU .GT. 0) GO TO 90
0181      C
0182      C SPACE FOR CALL WITH NO INTERACTION
0183      C
0184      C
0185      C INTERACTIVE CALL
0186      C
0187      90      IF (IBUF .NE. 1) GO TO 100
0188      C
0189      C GET DISC FILE NAME
0190      C
0191      WRITE(LU,1)
0192      1      FORMAT("ENTER DISC FILE NAME?_")
0193      READ(LU,2) NAME
0194      2      FORMAT(3A2)
0195      C
0196      C OPEN FILE
0197      C
0198      CALL OPEN(IDCB,IERR,NAME,0,0,0,1040)
0199      IF (IERR .LT. 0) GO TO 999
0200      WRITE(LU,3)
0201      3      FORMAT(" DATA FORMAT (1=UNPACKED; 2=PACKED; 3=REAL)?_")
0202      READ(LU,*) IFMT
0203      PACKED = .TRUE.
0204      IF (IFMT .NE. 2) PACKED = .FALSE.
0205      NUM = NPIXL
0206      IF (PACKED) NUM = (NPIXL+1)/2
0207      IF (IFMT .EQ. 3) NUM = 2*NPIXL
0208      IBCOD = 1
0209      RETURN
0210      C
0211      100     IF(IBUF .NE. 2) GO TO 110
0212      C
0213      C TAPE INPUT
0214      C
0215      WRITE(LU,4)
0216      4      FORMAT("TAPE LU?_")
0217      READ(LU,*) MTLU
0218      C
```

```
0219 C REWIND TAPE
0220 C
0221     CALL EXEC(3,MTLU+400B)
0222     WRITE(LU,9)
0223 9     FORMAT(" FILE #?_")
0224     READ(LU,*) IFILE
0225     IF (IFILE .LE. 0) CALL EXEC(6)
0226     IF (IFILE .EQ. 1) GO TO 107
0227     DO 105 I=1,IFILE-1
0228     CALL EXEC(3,MTLU+1300B)
0229 105    CONTINUE
0230 C
0231 107    WRITE(LU,3)
0232    READ(LU,*) IFMT
0233    PACKED = .TRUE.
0234    IF (IFMT .NE. 2) PACKED = .FALSE.
0235    NUM = NPIXL
0236    IF (PACKED) NUM = (NPIXL+1)/2
0237    IF (IFMT .EQ. 3) NUM = 2*NPIXL
0238    IBCOD = 2
0239    RETURN
0240 C
0241 110    IF (IBUF .NE. 3) GO TO 115
0242 C
0243 C DISPLAY INPUT
0244 C
0245     WRITE(LU,5)
0246 5     FORMAT("ENTER START LINE,END LINE,START PIXEL,END PIXEL?_")
0247     READ(LU,*) ISTRTL,IENDL,ISTRTP,IENDP
0248     ISTEP = 1
0249     IF (ISTRTL .GT. IENDL) ISTEP = -1
0250     PACKED = .FALSE.
0251     NUM = NPIXL
0252     IBCOD = 3
0253     RETURN
0254 C
0255 C INPUT IS WORK FILE
0256 C
0257 115    CALL OPEN(IDCB,IERR,6HWF0000,0,0,0,1040)
0258     IF (IERR .LT. 0) GO TO 999
0259     PACKED = .FALSE.
0260     NUM = 2*NPIXL
0261     IBCOD = 1
0262 C
0263 C POSITION FILE
0264 C
0265     CALL READF(IDCB,IERR,IData,0)
0266     IF (IERR .LT. 0) GO TO 999
0267 C
0268     RETURN
0269 C
0270 C
0271 C DATA INPUT SECTION
0272 C
0273 C BRANCH TO APPROPRIATE SUB SECTION
0274 C
```

```
0275 C
0276 120 GO TO (130,140,150),IBCOD
0277 C
0278 C FILE INPUT
0279 C
0280 130 CALL READF(IDCB,IERR,RDATA,NUM)
0281 IF (IERR .LT. 0) GO TO 999
0282 IFMT=3
0283 GO TO 160
0284 C
0285 C TAPE INPUT
0286 C
0287 140 CALL EXEC(1,MTLU,IData,NUM)
0288 GO TO 160
0289 C
0290 C DISPLAY INPUT
0291 C
0292 150 IBUF = 0
0293 CALL MVW(IBUF,IBUF(2),NPIXL-1)
0294 IF ((ISTEP .GT. 0) .AND.(ISTRRL .GT. IENDL)) RETURN
0295 IF (ISTEP .LT. 0 .AND. ISTRRL .LT. IENDL) RETURN
0296 CALL RLINE(ISTRRL,ISTRTP,IENDP,IData)
0297 ISTRRL = ISTRRL + ISTEP
0298 C
0299 C MOVE DATA TO OUTPUT ARRAY AND UNPACK IF NECESSARY
0300 C
0301 160 IF (.NOT. PACKED) GO TO 180
0302 C
0303 C DATA IN PACKED FORMAT
0304 C
0305 DO 170 I=1,NUM
0306 ITEMP = IData(I)
0307 CALL ROT8(ITEMP,JTEMP)
0308 JTEMP = IAND(JTEMP,377B)
0309 IF (JTEMP .GT. IP1) IP1 = JTEMP
0310 IF (JTEMP .LT. IP2) IP2 = JTEMP
0311 ITEMP = IAND(ITEMP,377B)
0312 IF (ITEMP .GT. IP1) IP1 = ITEMP
0313 IF (ITEMP .LT. IP2) IP2 = ITEMP
0314 IBUF(2*I-1) = JTEMP
0315 170 IBUF(2*I) = ITEMP
0316 RETURN
0317 C
0318 C DATA IS UNPACKED
0319 C
0320 180 DO 190 I=1,NPIXL
0321 ITEMP = IData(I)
0322 IF (IFMT .EQ. 3) ITEMP = RDATA(I)
0323 IF (ITEMP .GT. IP1) IP1 = ITEMP
0324 IF (ITEMP .LT. IP2) IP2 = ITEMP
0325 190 IBUF(I) = ITEMP
0326 RETURN
0327 C
0328 999 RDREC = IERR
0329 END
0330 $
0331 $
```

SLFLTR T=00003 IS ON CR00022 USING 00024 BLKS R=0000

```

0001  FTN4,L
0002      PROGRAM LFLTR
0003  C
0004  C      WRITTEN BY E. E. SHERROD
0005  C
0006  C      PROGRAM DOES LINEAR FILTERING USING SPATIAL DOMAIN
0007  C      RECURSIVE DIGITAL FILTERS
0008  C
0009  C
0010  C
0011  C
0012  C
0013      DIMENSION A(3,3,2),B(3,3,2),ILU(5),SUM(3,2)
0014      DIMENSION F1(524),F2(524),F3(524)
0015      DIMENSION G1(1),G2(1),G3(1),IX1(3)
0016      DIMENSION X1(524),X2(524),X3(524)
0017      DIMENSION IDCB(144),NAME(3),IRTN(5)
0018      COMMON /IBLK/IBUF(80)
0019      INTEGER READL,RITEL,WFINT
0020      EQUIVALENCE(IBUF(1),A(1,1,1)),(IBUF(41),B(1,1,1))
0021      EQUIVALENCE(IRTN(2),RMAX),(IRTN(4),RMIN)
0022      DATA NAME/2HCO,2HEF,2HFS/
0023  C
0024  C      NROW X 512 IMAGE
0025  C
0026      CALL RMPAR(ILU)
0027  C
0028      LU=ILU(1)
0029      IPIXL=ILU(2)
0030      JPIXL=ILU(3)
0031  C
0032  C      GET FILTER COEFF'S
0033      CALL OPEN(IDCB,IERR,NAME)
0034      IF(IERR .LT. 0) GO TO 9999
0035      CALL READF(IDCB,IERR,IBUF,80,IERR)
0036      IF(IERR .LT. 0) GO TO 9999
0037      NSTAG = ILUF(40)
0038      N = NSTAG + 1
0039      CALL CLOSE(IDCB,IERR)
0040  C
0041  C      GET CONTROL BLOCK INFORMATION
0042  C
0043      IERR=WFINT(NROW,ICOLS,RMAX,RMIN,LU)
0044      IF(IERR .LT. 0) GOTO 9999
0045      IPIXL = 2
0046      ICOLS=ICOLS-2
0047      JPIXL =ICOLS - 1
0048  C

```

```

0049 C
0050 C      INITIALIZE FILTER TO MID LINE-COL AVG
0051 C
0052      NMID=NROW/2
0053      CNST=0.0
0054      IERR=READL(NMID,0,511,F1)
0055      IF(IERR .LT. 0) GO TO 9999
0056 701      DO 110 I=1,ICOLS
0057 110      CNST=CNST+F1(I)
0058 602      CNST=CNST/FLOAT(ICOLS)
0059 C
0060      DO 13 I=1,524
0061      F3(I)=CNST
0062      F2(I)=CNST
0063 13      F1(I)=CNST
0064 C
0065 C      CALCULATE FINAL VALUE FOR EACH STAGE
0066 C
0067      DO 10 NSTG=2,N
0068      SUM(NSTG,1)=0.0
0069      SUM(NSTG,2)=0.0
0070      DO 11 I=1,3
0071      DO 11 J=1,3
0072      SUM(NSTG,1)=SUM(NSTG,1)+A(I,J,NSTG-1)
0073 11      SUM(NSTG,2)=SUM(NSTG,2)+B(I,J,NSTG-1)
0074      DEL=ABS(SUM(NSTG,2))
0075      IF(DEL.LT.1.0E-20)CALL EXEC(2,LU,16HFILTER UNSTABLE ,8)
0076 10      SUM(NSTG,1)=SUM(NSTG,1)/SUM(NSTG,2)
0077 C
0078 C      CALCULATE INITIAL CONDITIONS FOR EACH STAGE
0079 C
0080      SUM(1,2)=CNST
0081      DO 12 NSTG=2,N
0082 12      SUM(NSTG,2)=SUM(NSTG,1)*SUM(NSTG-1,2)
0083 C
0084 C      INITIALIZE FILTER
0085 C
0086      DO 14 I=1,524
0087      X3(I)=SUM(2,2)
0088      X2(I)=SUM(2,2)
0089      X1(I)=SUM(2,2)
0090      IF (NSTAG .EQ. 1) GO TO 14
0091      G3(I) = SUM(3,2)
0092      G2(I)=SUM(3,2)
0093      G1(I)=SUM(3,2)
0094 14      CONTINUE
0095      RMX=-1.0E38
0096      RMI= 1.0E38
0097 C
0098 C      FILTER REVERSE
0099 C
0100      IERR=READL(8,IPIXL,JPIXL,F3)
0101      IF(IERR .LT. 0) GO TO 9999
0102      IERR=READL(7,IPIXL,JPIXL,F2)
0103      IF(IERR .LT. 0) GO TO 9999
0104      IERR=READL(6,IPIXL,JPIXL,F1)
0105      IF(IERR .LT. 0) GO TO 9999

```

```

0106 C
0107     LNCK = 1
0108     DO 300 NRO=-6,NROW - 1,3
0109     CALL FILTR(2,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
0110     IF(LNCK .LT. 7) GO TO 301
0111     LINE = LABS(NRO)
0112     CALL RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,2,LU,RMX,RMI)
0113 301     LNCK =LNCK +1
0114     LINE=LABS(NRO+1)
0115     IF(LINE .GT. NROW-1) GO TO 300
0116     IERR=READL(LINE,IPIXL,JPIXL,F3)
0117     IF(IERR .LT. 0) GO TO 9999
0118     CALL FILTR(2,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
0119     IF(LNCK .LT. 7) GO TO 302
0120     CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,2,LU,RMX,RMI)
0121 302     LNCK =LNCK +1
0122     LINE=LABS(NRO+2)
0123     IF(LINE .GT. NROW-1) GO TO 300
0124     IERR=READL(LINE,IPIXL,JPIXL,F2)
0125     IF(IERR .LT. 0) GO TO 9999
0126     CALL FILTR(2,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
0127     IF(LNCK .LT. 7) GO TO 303
0128     IF(LINE .GT. NROW-1) GO TO 300
0129     CALL RITLN(LINE,IPIXL,JPIXL,X2,G1,NSTAG,2,LU,RMX,RMI)
0130 303     LNCK =LNCK +1
0131     LINE=LABS(NRO+3)
0132     IF(LINE .GT. NROW-1) GO TO 300
0133     IERR=READL(LINE,IPIXL,JPIXL,F1)
0134     IF(IERR .LT. 0) GO TO 9999
0135 300     CONTINUE
0136 C
0137 C REINITIALIZE FILTER
0138 C
0139     CONST=(RMX-RMI)/2.
0140     DO 15 II=1,524
0141     F1(II) = CONST
0142     F2(II) = CONST
0143     F3(II) = CONST
0144 15     CONTINUE
0145 C
0146 C     FILTER FORWARD
0147 C
0148     RMX=-0.1E38
0149     RMI= 0.1E38
0150     LINE =NROW-9
0151     IERR=READL(LINE,IPIXL,JPIXL,F3(12))
0152     IF(IERR .LT. 0) GO TO 9999
0153     LINE=LINE+1
0154     IERR=READL(LINE,IPIXL,JPIXL,F2(12))
0155     IF(IERR .LT. 0) GO TO 9999
0156     LINE=LINE+1
0157     IERR=READL(LINE,IPIXL,JPIXL,F1(12))
0158     IF(IERR .LT. 0) GO TO 9999

```

```

0159 C
0160      LNCK ==6
0161      DO 400 NRO= -6,NROW - 1,3
0162      CALL FILTR(1,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
0163      IF(LNCK .LT. 0) GO TO 401
0164      CALL RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,1,LU, RMX,RMI)
0165 401      LNCK=LNCK+1
0166      LINE=(NROW-1)-IABS(NRO+1)
0167      IERR=READL(LINE,IPIXL,JPIXL,F3(12))
0168      IF(IERR .LT. 0) GO TO 9999
0169      CALL FILTR(1,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
0170      IF(LNCK .LT. 0) GO TO 402
0171      CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,1,LU, RMX,RMI)
0172 402      LNCK =LNCK +1
0173      LINE=(NROW-1)-IABS(NRO+2)
0174      IF(LINE .LT. 0) GO TO 400
0175      IERR=READL(LINE,IPIXL,JPIXL,F2(12))
0176      IF(IERR .LT. 0) GO TO 9999
0177      CALL FILTR(1,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
0178      IF(LNCK .LT. 0) GO TO 403
0179      CALL RITLN(LINE,IPIXL,JPIXL,X2,G1,NSTAG,1,LU, RMX,RMI)
0180 403      LNCK =LNCK +1
0181      LINE=(NROW-1)-IABS(NRO+3)
0182      IF(LINE .LT. 0) GO TO 400
0183      IERR=READL(LINE,IPIXL,JPIXL,F1(12))
0184      IF(IERR .LT. 0) GO TO 9999
0185 400      CONTINUE
0186 C
0187 51      CONTINUE
0188      RMAX=RMX
0189      RMIN=RMI
0190      CALL CLSWF(NROW,ICOLS,RMAX,RMIN)
0191      CALL PRTN(IRTN)
0192      CALL EXEC(6)
0193 9999      CALL EXEC(2,LU,16HREAD FILE ERROR ,8)
0194      END
0195      SUBROUTINE FILTR(IFLAG,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
0196      DIMENSION F1(1),F2(1),F3(1),X1(1),X2(1),X3(1),A(1),B(1)
0197      COMMON /IBLK/IBUF(80)
0198      DIMENSION G1(1),G2(1),G3(1)
0199 C
0200      EQUIVALENCE (IBUF,A),(IBUF(41),B)
0201 C      IFLAG =1 FOR FORWARD FILTERING, = 2 FOR REVERSE
0202 C
0203 C REVERSE FILTERING
0204 C
0205      IF(IFLAG .EQ. 1) GO TO 200
0206      DO 20 I=1,11
0207      L =ICOLS+12 - I
0208      J = ICOLS-12 + I
0209      F1(L) = F1(J)
0210      F2(L) = F2(J)
0211 20      F3(L) = F3(J)
0212 C

```

```

0213      DO 10 M = ICOLS+9,1,-1
0214      J = M + 1
0215      K = M +2
0216      X1(M) = A(1) * F1(M)
0217      1      + A(2) * F1(J)-B(2)*X1(J)
0218      1      + A(3) * F1(K)-B(3)*X1(K)
0219      1      + A(4) * F2(M)-B(4)*X2(M)
0220      1      + A(5) * F2(J)-B(5) *X2(J)
0221      1      + A(6) * F2(K)-B(6) *X2(K)
0222      1      + A(7) * F3(M)-B(7)*X3(M)
0223      1      + A(8) * F3(J)-B(8) *X3(J)
0224      1      + A(9) * F3(K) - B(9)*X3(K)
0225      IF(NSTAG .EQ. 1) GO TO 10
C      G1(M) = A(10) * X1(M)
0227      1      + A(11) * X1(J)-B(11)*G1(J)
0228      1      + A(12) * X1(K)-B(12)*G1(K)
0229      1      + A(13) * X2(M)-B(13)*G2(M)
0230      1      + A(14) * X2(J)-B(14) *G2(J)
0231      1      + A(15) * X2(K)-B(15) *G2(K)
0232      1      + A(16) * X3(M)-B(16)*G3(M)
0233      1      + A(17) * X3(J)-B(17) *G3(J)
0234      1      + A(18) * X3(K) - B(18)*G3(K)
0235      10     CONTINUE
0236      GO TO 400
0237      200    CONTINUE
0238      C
0239      C FORWARD FILTERING
0240      C
0241      DO 30 I=1,11
0242      L =12 - I
0243      J = 12 + I
0244      F1(L) = F1(J)
0245      F2(L) = F2(J)
0246      30     F3(L) = F3(J)
0247      C
0248      DO 40 M = 3,ICOLS + 11
0249      J = M - 1
0250      K = M -2
0251      X1(M) = A(1) * F1(M)
0252      1      + A(2) * F1(J)-B(2)*X1(J)
0253      1      + A(3) * F1(K)-B(3)*X1(K)
0254      1      + A(4) * F2(M)-B(4)*X2(M)
0255      1      + A(5) * F2(J)-B(5) *X2(J)
0256      1      + A(6) * F2(K)-B(6) *X2(K)
0257      1      + A(7) * F3(M)-B(7)*X3(M)
0258      1      + A(8) * F3(J)-B(8) *X3(J)
0259      1      + A(9) * F3(K) - B(9)*X3(K)
0260      IF(NSTAG .EQ. 1) GO TO 40
0261      G1(M) = A(10) * X1(M)
0262      1      + A(11) * X1(J)-B(11)*G1(J)
0263      1      + A(12) * X1(K)-B(12)*G1(K)
0264      1      + A(13) * X2(M)-B(13)*G2(M)
0265      1      + A(14) * X2(J)-B(14) *G2(J)
0266      1      + A(15) * X2(K)-B(15) *G2(K)
0267      1      + A(16) * X3(M)-B(16)*G3(M)
0268      1      + A(17) * X3(J)-B(17) *G3(J)
0269      1      + A(18) * X3(K) - B(18)*G3(K)
0270      40     CONTINUE
0271      400   CONTINUE
0272      RETURN
0273      END

```

```
0274 C
0275 C COMMON BLOCK SUBPROGRAM
0276 C
0277     BLOCK DATA IBLK
0278     COMMON /IBLK/IBUF(80)
0279     DATA IBUF/80*0/
0280     END
0281     SUBROUTINE RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,IFLAG,LU, RMX,R
0282             DIMENSION X1(1),G1(1),IX1(524)
0283     INTEGER RITEL
0284     IFL=1
0285     IF(IFLAG .EQ. 1) IFL = 12
0286     IF(NSTAG .EQ. 2) GO TO 100
0287     IERR= RITEL(LINE,IPIXL,JPIXL,X1(IFL))
0288 12     IF(IERR .LT. 0) GO TO 9999
0289     DO 120 I=IFL,JPIXL-IPIXL +IFL
0290     IF(X1(I) .GT. RMX) RMX=X1(I)
0291     IF(X1(I) .LT. RMI) RMI=X1(I)
0292     ITEMP= X1(I) + 0.5
0293     IF(ITEMP .LT. 0) ITEMP=0
0294     IF(ITEMP .GT. 377B) ITEMP=377B
0295 120    IX1(I) = ITEMP
0296     GO TO 200
0297 100    CONTINUE
0298     IERR= RITEL(LINE,IPIXL,JPIXL,G1(IFL))
0299     IF(IERR .LT. 0) GO TO 9999
0300     DO 121 I=1,524
0301     ITEMP=G1(I) + 0.5
0302     IF(ITEMP .LT. 0) ITEMP=0
0303 121    IX1(I) =IAND(ITEMP,777B)
0304 200    ISTRT=(511-JPIXL)/2
0305    ISTOP=ISTRT+JPIXL
0306        CALL WLINE(LINE,ISTRT,ISTOP,IX1(IFL))
0307        RETURN
0308 9999    CALL EXEC(2,LU,16HWRITE FILE ERROR,8)
0309        END
0310    END$
```

SHFLTR T=00004 IS ON CRO0022 USING 00036 BLKS R=0289

```

0001  FTN4,L
0002      PROGRAM HFLTR
0003  C
0004  C      WRITTEN BY E. E. SHERROD
0005  C
0006  C      PROGRAM DOES HOMOMORPHIC FILTERING USING SPATIAL DOMAIN
0007  C      Recursive DIGITAL FILTERS
0008  C
0009      COMMON /IBLK/IBUF(80)
0010      DIMENSION IF1(2),IF2(523),R1(523)
0011          DIMENSION A(3,3,2),B(3,3,2),ILU(5),SUM(3,2)
0012          DIMENSION F1(523),F2(523),F3(523)
0013          DIMENSION G1(1),G2(1),G3(1),IX1(3)
0014          DIMENSION X1(523),X2(523),X3(523)
0015          DIMENSION IDCDB(144),NAME(3),IRTN(5)
0016      INTEGER READL,RITEL,WFINT
0017          EQUIVALENCE(IBUF(1),A(1,1,1)),(IBUF(41),B(1,1,1))
0018          EQUIVALENCE(IRTN(2),RMAX),(IRTN(4),RMIN)
0019          EQUIVALENCE(F1,R1),(F2,IF2),(R1,IF1),(IF1,ILINE),(IF1(2),ICO
0020 1(R1(2),RMAXX),(R1(3),RMINN)
0021          DATA NAME/2HCO,2HEF,2HFS/
0022  C
0023      CALL RMPAR(ILU)
0024      LU=ILU(1)
0025      IF(LU .EQ. 0) LU=1
0026      IPIXL = ILU(2)
0027      IF(IPIXL .EQ. 0) IPIXL = 0
0028      JPIXL = ILU(3)
0029      IF(JPIXL .EQ. 0) JPIXL = 511
0030  C
0031  C GFT FILTER COEFF'S
0032      CALL OPEN(IDCB,IERR,NAME)
0033      IF(IERR .LT. 0) GO TO 9999
0034      CALL READF(IDCB,IERR,IBUF,80,IERR)
0035      IF(IERR .LT. 0) GO TO 9999
0036      NSTAG = IBUF(40)
0037      N = NSTAG + 1
0038      CALL CLOSE(IDCB,IERR)
0039  C
0040  C GET CONTROL BLOCK INFORMATION
0041      IERR=WFINT(NROW,ICOLS,RMAX,RMIN,LU)
0042      IF(IERR .LT. 0)GOTO 9999
0043      IPIXL=2
0044      ICOLS=ICOLS-2
0045      JPIXL = ICOLS -1
0046  C
0047  C      INITIALIZE FILTER TO MID LINE-COL AVG
0048      NMID=NROW/2
0049      CNST=0.0
0050      IERR=READL(NMID,IPIXL,JPIXL,F1)
0051      IF(IERR .LT. 0) GO TO 9999
0052      CALL BLAS(F1,RMIN,ICOLS)
0053  701      DO 110 I=1,ICOLS
0054  110      CNST=CNST+AMAX0(F1(I),1)
0055  602      CNST=(CNST/FLOAT(ICOLS))
0056      CNST = ALOG(CNST)

```

```

0057  C
0058      DO 9 I=1,523
0059      F1(I) = CNST
0060      F2(I) = CNST
0061      F3(I) = CNST
0062  9      CONTINUE
0063  C
0064  C      CALCULATE FINAL VALUE FOR EACH STAGE
0065      DO 10 NSTG=2,N
0066      SUM(NSTG,1)=0.0
0067      SUM(NSTG,2)=0.0
0068      DO 11 I=1,3
0069      DO 11 J=1,3
0070      SUM(NSTG,1)=SUM(NSTG,1)+A(I,J,NSTG-1)
0071  11      SUM(NSTG,2)=SUM(NSTG,2)+B(I,J,NSTG-1)
0072      DEL=ABS(SUM(NSTG,2))
0073      IF(DEL.LT.1.0E-20)CALL EXEC(2,LU,16HFILTER UNSTABLE ,8)
0074  10      SUM(NSTG,1)=SUM(NSTG,1)/SUM(NSTG,2)
0075  C
0076  C      CALCULATE INITIAL CONDITIONS FOR EACH STAGE
0077      SUM(1,2)=CNST
0078      DO 12 NSTG=2,N
0079  12      SUM(NSTG,2)=SUM(NSTG,1)*SUM(NSTG-1,2)
0080  C
0081  C      INITIALIZE FILTER
0082      DO 14 I=1,523
0083      X3(I)=(SUM(2,2))
0084      X2(I)=(SUM(2,2))
0085      X1(I)=(SUM(2,2))
0086      IF (NSTAG .EQ. 1) GO TO 14
0087      G3(I) =( SUM(3,2))
0088      G2(I) =( SUM(3,2))
0089      G1(I) =( SUM(3,2))
0090  14      CONTINUE
0091      RMX=-1.0E38
0092      RMI= 1.0E38
0093  C
0094  C      FILTER REVERSE
0095      CALL EXEC(2,LU,16HREVERSE FILTERIN,8)
0096      SCL = 1.0
0097      IERR=READL(8,IPIXL,JPIXL,F3)
0098      IF(IERR .LT. 0) GO TO 9999
0099      CALL BIAS(F3,RMIN,ICOLS)
0100      IERR=READL(7,IPIXL,JPIXL,F2)
0101      IF(IERR .LT. 0) GO TO 9999
0102      CALL BIAS(F2,RMIN,ICOLS)
0103      IERR=READL(6,IPIXL,JPIXL,F1)
0104      IF(IERR .LT. 0) GO TO 9999
0105  C

```

```

0106      LNCK = 1
0107      DO 300 NRO=-6,NROW - 1,3
0108      CALL BIAS(F1,RMIN,ICOLS)
0109      CALL HFILT(2,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
0110      IF(LNCK .LT. 7) GO TO 301
0111      LINE = LABS(NRO)
0112      CALL RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,2,LU,RMX,RMI,SCL)
0113 301      LNCK =LNCK +1
0114      LINE=LABS(NRO+1)
0115      IF(LINE .GT. NROW-1) GO TO 300
0116      IERR=READL(LINE,IPIXL,JPIXL,F3)
0117      IF(IERR .LT. 0) GO TO 9999
0118      CALL BIAS(F3,RMIN,ICOLS)
0119      CALL HFILT(2,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
0120      IF(LNCK .LT. 7) GO TO 302
0121      CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,2,LU,RMX,RMI,SCL)
0122 302      LNCK =LNCK +1
0123      LINE=LABS(NRO+2)
0124      IF(LINE .GT. NROW-1) GO TO 300
0125      IERR=READL(LINE,IPIXL,JPIXL,F2)
0126      IF(IERR .LT. 0) GO TO 9999
0127      CALL BIAS(F2,RMIN,ICOLS)
0128      CALL HFILT(2,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
0129      IF(LNCK .LT. 7) GO TO 303
0130      IF(LINE .GT. NROW-1) GO TO 300
0131      CALL RITLN(LINE,IPIXL,JPIXL,X2,G1,NSTAG,2,LU,RMX,RMI,SCL)
0132 303      LNCK =LNCK +1
0133      LINE=LABS(NRO+3)
0134      IF(LINE .GT. NROW-1) GO TO 300
0135      IERR=READL(LINE,IPIXL,JPIXL,F1)
0136      IF(IERR .LT. 0) GO TO 9999
0137 300      CONTINUE
0138 C
0139 C REINITIALIZE FILTER
0140      CONST = (RMX-RMI)/2.
0141      DO 15 J=1,523
0142      F1(J) = CONST
0143      F2(J) = CONST
0144      F3(J) = CONST
0145 15      CONTINUE
0146 C
0147 C      FILTER FORWARD
0148 C
0149      CALL EXEC(2,LU,16HFORWARD FILTERIN,8)
0150 C
0151 C SCALE FOR LN(32766)
0152      SCL = 10.397147 /(RMX)
0153      RMI=0.1E38
0154      RMX=-0.1E38
0155      JPIXL=JPIXL-1
0156      LINE =NROW-9
0157      IERR=READL(LINE,IPIXL,JPIXL,F3(12))
0158      IF(IERR .LT. 0) GO TO 9999
0159      LINE=LINE+1
0160      IERR=READL(LINE,IPIXL,JPIXL,F2(12))
0161      IF(IERR .LT. 0) GO TO 9999
0162      LINE=LINE+1
0163      IERR=READL(LINE,IPIXL,JPIXL,F1(12))
0164      IF(IERR .LT. 0) GO TO 9999

```

```

0165 C
0166 LNCK --6
0167 DO 400 NRO= -6,NROW - 1,3
0168 CALL HFILT(1,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
0169 IF(LNCK .LT. 0) GO TO 401
0170 CALL RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,1,LU, RMX,RMI,SCL)
0171 401 LNCK=LNCK+1
0172 LINE=(NROW-1)-ABS(NRO+1)
0173 IERR=READL(LINE,IPIXL,JPIXL,F3(12))
0174 IF(IERR .LT. 0) GO TO 9999
0175 CALL HFILT(1,F3,F1,F2,X3,X1,X2,G1,NSTAG,ICOLS)
0176 IF(LNCK .LT. 0) GO TO 402
0177 CALL RITLN(LINE,IPIXL,JPIXL,X3,G1,NSTAG,1,LU, RMX,RMI,SCL)
0178 402 LNCK =LNCK +1
0179 LINE=(NROW-1)-ABS(NRO+2)
0180 IF(LINE .LT. 0) GO TO 400
0181 IERR=READL(LINE,IPIXL,JPIXL,F2(12))
0182 IF(IERR .LT. 0) GO TO 9999
0183 CALL HFILT(1,F2,F3,F1,X2,X3,X1,G1,NSTAG,ICOLS)
0184 IF(LNCK .LT. 0) GO TO 403
0185 CALL RITLN(LINE,IPIXL,JPIXL,X2,G1,NSTAG,1,LU, RMX,RMI,SCL)
0186 403 LNCK =LNCK +1
0187 LINE=(NROW-1)-ABS(NRO+3)
0188 IF(LINE .LT. 0) GO TO 400
0189 IERR=READL(LINE,IPIXL,JPIXL,F1(12))
0190 IF(IERR .LT. 0) GO TO 9999
0191 400 CONTINUE
0192 C
0193 51 CONTINUE
0194 CALL EXEC(2,LU,10HCOMPLETED ,5)
0195 C
0196 CALL CLSWF(NROW,ICOLS, RMX,RMI)
0197 C
0198 RMAX = RMX
0199 RMIN = RMI
0200 CALL PRTN(IRTN)
0201 CALL EXEC(6)
0202 9999 CALL EXEC(2,LU,16HREAD FILE ERROR ,8)
0203 END
0204 SUBROUTINE HFILT(IFLAG,F1,F2,F3,X1,X2,X3,G1,NSTAG,ICOLS)
0205 DIMENSION F1(1),F2(1),F3(1),X1(1),X2(1),X3(1),A(1),B(1)
0206 COMMON /IBLK/IBUF(80)
0207 DIMENSION G1(1),G2(1),G3(1)
0208 C
0209 EQUIVALENCE (IBUF,A),(IBUF(41),B)
0210 C IFLAG =1 FOR FORWARD FILTERING, = 2 FOR REVERSE
0211 C
0212 C REVERSE FILTERING
0213 C
0214 IF(IFLAG .EQ. 1) GO TO 200
0215 DO 20 I=1,11
0216 L =ICOLS+12 - I
0217 J = ICOLS-12 + I
0218 F1(L) = F1(J)
0219 F2(L) = F2(J)
0220 20 F3(L) = F3(J)
0221 C

```

```

0222      DO 10 M = ICOLS+9,1,-1
0223      J = M + 1
0224      K = M +2
0225      X1(M) = A(1) * ALOG(F1(M))
0226      1      + A(2) * ALOG(F1(J))-B(2)*X1(J)
0227      1      + A(3) * ALOG(F1(K))-B(3)*X1(K)
0228      1      + A(4) * ALOG(F2(M))-B(4)*X2(M)
0229      1      + A(5) * ALOG(F2(J))-B(5) *X2(J)
0230      1      + A(6) * ALOG(F2(K))-B(6) *X2(K)
0231      1      + A(7) * ALOG(F3(M))-B(7)*X3(M)
0232      1      + A(8) * ALOG(F3(J))-B(8) *X3(J)
0233      1      + A(9) * ALOG(F3(K))- B(9)*X3(K)
0234      IF(NSTAG .EQ. 1) GO TO 10
0235      G1(M) = A(10) * X1(M)
0236      1      + A(11) * X1(J)-B(11)*G1(J)
0237      1      + A(12) * X1(K)-B(12)*G1(K)
0238      1      + A(13) * X2(M)-B(13)*G2(M)
0239      1      + A(14) * X2(J)-B(14) *G2(J)
0240      1      + A(15) * X2(K)-B(15) *G2(K)
0241      1      + A(16) * X3(M)-B(16)*G3(M)
0242      1      + A(17) * X3(J)-B(17) *G3(J)
0243      1      + A(18) * X3(K) - B(18)*G3(K)
0244      10     CONTINUE
0245      GO TO 400
0246      200    CONTINUE
0247      C
0248      C FORWARD FILTERING
0249      C
0250      DO 30 I=1,11
0251      L =12 - I
0252      J = 12 + I
0253      F1(L) = F1(J)
0254      F2(L) = F2(J)
0255      30     F3(L) = F3(J)
0256      C
0257      DO 40 M = 3,ICOLS+11
0258      J = M - 1
0259      K = M -2
0260      X1(M) = A(1) * F1(M)
0261      1      + A(2) * F1(J)-B(2)*X1(J)
0262      1      + A(3) * F1(K)-B(3)*X1(K)
0263      1      + A(4) * F2(M)-B(4)*X2(M)
0264      1      + A(5) * F2(J)-B(5) *X2(J)
0265      1      + A(6) * F2(K)-B(6) *X2(K)
0266      1      + A(7) * F3(M)-B(7)*X3(M)
0267      1      + A(8) * F3(J)-B(8) *X3(J)
0268      1      + A(9) * F3(K) - B(9)*X3(K)
0269      IF(NSTAG .EQ. 1) GO TO 40
0270      G1(M) = A(10) * X1(M)
0271      1      + A(11) * X1(J)-B(11)*G1(J)
0272      1      + A(12) * X1(K)-B(12)*G1(K)
0273      1      + A(13) * X2(M)-B(13)*G2(M)
0274      1      + A(14) * X2(J)-B(14) *G2(J)
0275      1      + A(15) * X2(K)-B(15) *G2(K)
0276      1      + A(16) * X3(M)-B(16)*G3(M)
0277      1      + A(17) * X3(J)-B(17) *G3(J)
0278      1      + A(18) * X3(K) - B(18)*G3(K)
0279      40     CONTINUE
0280      400    CONTINUE
0281      RETURN
0282      END

```

```

0283 C
0284 C  COMMON BLOCK SUBPROGRAM
0285 C
0286     BLOCK DATA IBLK
0287     COMMON/IBLK/IBUF(80)
0288     END
0289     SUBROUTINE RITLN(LINE,IPIXL,JPIXL,X1,G1,NSTAG,IFLAG,LU,RMX,R
0290     SCL)
0291     DIMENSION X1(1),G1(1),XX1(523)
0292     INTEGER RITEL
0293 C
0294 C IFLAG =1 FOR FORWARD  =2 FOR REVERSE
0295 C REV
0296     IF(NSTAG .EQ. 2) GO TO 12
0297     IF(IFLAG .EQ. 1) GO TO 11
0298     DO 10 M=1,JPIXL-IPIXL+1
0299     IF(X1(M) .GT. RMX) RMX=X1(M)
0300     IF(X1(M) .LT. RMI) RMI=X1(M)
0301 10     CONTINUE
0302     IERR=RITEL(LINE,IPIXL,JPIXL,X1)
0303     IF(IERR .LT. 0) GO TO 9999
0304     GO TO 12
0305 C
0306 11     CONTINUE
0307 C FORWARD
0308     DO 20 M=12,JPIXL-IPIXL+12
0309     X=SCL*(X1(M))
0310     IF(X .GT. 10.397147) X = 10.397177
0311     XX1(M) = EXP(X)
0312     IF(XX1(M) .GT. RMX) RMX=XX1(M)
0313     IF(XX1(M) .LT. RMI) RMI=XX1(M)
0314 20     CONTINUE
0315     IERR= RITEL(LINE,IPIXL,JPIXL,XX1(12))
0316     IF(IERR .LT. 0) GO TO 9999
0317 12     CONTINUE
0318     RETURN
0319 9999 CALL EXEC(2,LU,16HWRITE FILE ERROR,8)
0320     END
0321     SUBROUTINE BIAS(F1,RMIN,ICOLS)
0322     DIMENSION F1(1)
0323     DO 10 I=1,ICOLS + 11
0324     F1(I) = F1(I) - RMIN +1.0
0325     IF(F1(I) .LT. 1.) F1(I) = 1.0
0326 10     CONTINUE
0327     RETURN
0328     END
0329 $

```

&SHOW T=00004 IS ON CRO0022 USING 00005 BLKS R=0037

```
0001  FIN4
0002      PROGRAM SHOW
0003  C
0004      DIMENSION RDATA(512),IDATA(512),LU(5)
0005  C
0006      INTEGER READL
0007      EQUIVALENCE (RDATA,LU(2)),(LU(2),ILINE),(LU(3),IPIXL),
0008      1 (RDATA(2),RMAX),(RDATA(3),RMIN)
0009  C
0010  C GET INPUT PARAMETERS
0011  C
0012      CALL RMPAR(LU)
0013  C
0014  C GET SCALE
0015  C
0016      WRITE(LU,1)
0017  1      FORMAT("INPUT RANGE? _")
0018      READ(LU,*)RL,RH
0019  C
0020  C READ WORK FILE HEADER
0021  C
0022      IERR = READL(-1,0,511,RDATA)
0023      IF (IERR .LT. 0) GO TO 999
0024      NLINE = ILINE
0025      NPIXL = IPIXL
0026      PMAX = RMAX
0027      PMIN = RMIN
0028      DO 100 I=0,NLINE-1
0029      IF (READL(I,0,NPIXL-1,RDATA) .LT. 0) GO TO 999
0030      DO 90 J =1,NPIXL
0031      IDATA(J) = RL +((RH-RL)/(PMAX-PMIN))*(RDATA(J)-PMIN)
0032      IF (IDATA(J) .GT. 255) IDATA(J) = 255
0033      IF (IDATA(J) .LT. 0) IDATA(J) = 0
0034  90      CONTINUE
0035  C
0036      CALL WLINE(I,0,511,IDATA)
0037  100     CONTINUE
0038      CALL CLSWF(NLINE,NPIXL,PMAX,PMIN)
0039      CALL EXEC(6)
0040  999     WRITE(LU,2) IERR
0041  2      FORMAT("FILE ERROR",I7)
0042      END
0043  $
```

&FIRO T=00004 IS ON CR00022 USING 00003 BLKS R=0023

```

0001  FTN4,L
0002      PROGRAM FIRO
0003      DIMENSION ILU(5),IBUF(80),A(3,3,2),H(5,5),NAME(3),IDCB(144)
0004      DIMENSION NAME1(3),NAME2(3)
0005      EQUIVALENCE (IBUF(1),A(1,1,1))
0006      DATA H/25*0./
0007      DATA IBUF/80*0/
0008      DATA NAME/2HCO,2HEF,2HFS/
0009      DATA NAME1/2HDP,2HLA,2HM /
0010      DATA NAME2/2HPL,2HOT,2HV /
0011  C
0012  C GET LU
0013      CALL RMPAR(ILU)
0014      LU=ILU
0015      WRITE(LU,10)
0016  10      FORMAT(" ENTER NUMBER OF STAGES _")
0017      READ(LU,*) NSTG
0018      IBUF(40)=NSTG
0019  C
0020      WRITE(LU,11)
0021  11      FORMAT(" ENTER ALPHA VALUE _")
0022      READ(LU,*) ALPHA
0023  C
0024      H(1,1)=1.0
0025      DO 100 I=1,3
0026      DO 100 J=1,3
0027      CALL WINDO(ALPHA,I,J,WIN)
0028      A(I,J,NSTG)=WIN*H(I,J)
0029  100     CONTINUE
0030      CALL PURGE(IDCB,IERR,NAME,2HES)
0031      IF(IERR .LT. 0) WRITE(LU,999) IERR
0032      CALL CREAT(IDCB,IERR,NAME,2,3,2HES)
0033      IF(IERR .LT. 0) WRITE(LU,999) IERR
0034      CALL WRITF(IDCB,IERR,IBUF,80)
0035      CALL CLOSE(IDCB,IERR)
0036  C
0037  C SCHEDULE DISPLAY
0038      CALL EXEC(23,NAME1,LU,NSTG,0,0,0,IBUF,80)
0039  C
0040      WRITE(LU,40)
0041  40      FORMAT("//" ENTER DISPLAY DEVICE "//" 1. TV"/" 2. HP2648A")
0042      READ(LU,*) IDEV
0043      IF(IDEV .EQ. 2) GO TO 41
0044      CALL EXEC(23,NAME2)
0045      GO TO 42
0046  41      CONTINUE
0047      CALL HP48A(LU)
0048  42      CONTINUE
0049  999     FORMAT(" FILE ERROR ")
0050      STOP
0051      END

```

```

0052      SUBROUTINE HP48A(LU)
0053      DIMENSION IB(14),IA(4)
0054      INTEGER IDCB(144),BUFF( 4 ), NAME(3)
0055      DATA NAME/2HDA,2HTA,2H1 /
0056      C
0057      CALL OPEN(IDCB,IERR,NAME)
0058      IF (IERR .GE. 0) GO TO 30
0059      WRITE(LU,10) IERR
0060      10  FORMAT ("OPEN ERROR",F5.0)
0061      STOP
0062      30  CALL GRAFC(1,LU)
0063      20  CALL READF(IDCB,IERR,BUFF,4,ILOG)
0064      IF(ILOG .EQ. -1) GO TO 55
0065      IF (IERR .GE. 0) GOTO 40
0066      WRITE(LU,31) IERR
0067      31  FORMAT("READ ERROR",F5.0)
0068      GO TO 55
0069      40  CONTINUE
0070      CALL DVECT(BUFF,BUFF(2),BUFF(3),BUFF(4),LU)
0071      50  GO TO 20
0072      55  CALL EXEC(13,LU,ISTAT)
0073      ISTAT=LAND(ISTAT,140000B)
0074      IF(ISTAT.NE.0) GO TO 55
0075      CALL GRAFC(0,LU)
0076      CALL CLOSE(IDCB)
0077      RETURN
0078      END
0079      SUBROUTINE GRAFC(IFLAG,LU)
0080      INTEGER IESC
0081      IESC= 33B
0082      C
0083      C GRAPHICS OFF=0; GRAPHICS ON NOT=0
0084      C
0085      IF(IFLAG.EQ.0) GO TO 100
0086      C
0087      C GRAPHIC ON
0088      C
0089      WRITE(LU,10) IESC
0090      10  FORMAT(1R2,"*dC")
0091      WRITE(LU,12) IESC
0092      12  FORMAT(1R2,"*dF")
0093      WRITE(LU,14) IESC
0094      14  FORMAT(1R2,"*dA")
0095      C
0096      GO TO 200
0097      C
0098      C GRAPHICS OFF
0099      C
0100     100  WRITE(LU,30) IESC
0101     30  FORMAT(1R2,"*dd")
0102     WRITE(LU,40) IESC
0103     40  FORMAT(1R2,"*dE")
0104     200 RETURN
0105      END

```

```
0106      SUBROUTINE DVECT(IX1,IY1,IX2,IY2,LU)
0107  C
0108  C      SUBROUTINE DRAWS A LINE BETWEEN THE TWO POINTS (IX1,IY1)
0109  C          AND (IX2,IY2).  THE POINT (IX0,IY0) DEFINES THE
0110  C          THE ORIGIN.
0111  C
0112      IX0=0
0113      IY0=0
0114      XSCAL =356.0/1024.0
0115      YSCAL =XSCAL
0116      X1 = IX1*XSCAL + 0.5
0117      X2 = IX2*XSCAL + 0.5
0118      Y1 = IY1*YSCAL + 0.5
0119      Y2 = IY2*YSCAL + 0.5
0120      JX1 = X1 + IX0
0121      JX2 = X2 + IX0
0122      JY1 = Y1 + IY0
0123      JY2 = Y2 + IY0
0124      WRITE(LU,10) JX1,JY1,JX2,JY2
0125  10      FORMAT("pa",1I3,1H,,1I3,1H,,1I3,1H,,1I3,"z")
0126      RETURN
0127      END
0128      END$
```

0129 \$
0130 \$

&WINDO T=00004 IS ON CR00022 USING 00003 BLKS R=0012

```

0001  FTN4
0002      SUBROUTINE WINDO(ALPHA,N,M,WIN)
0003      XN=SQRT(M**2 + N**2)
0004      BETA=ALPHA*SQRT(1.-XN)
0005      CALL BESIO(ALPHA,BIAA)
0006      CALL BESIO(BETA,BIBB)
0007      BETA1=ALPHA*SQRT(2)
0008      CALL BESIO(BETA1,BIB)
0009      ZMIN=BIB/BIAA
0010      WIN=(BIBB/BIAA-ZMIN)/(1.0-ZMIN)
0011      RETURN
0012      END
0013  $
```

&BESIO T=00004 IS ON CR00022 USING 00002 BLKS R=0015

```

0001  FTN4
0002      SUBROUTINE BESIO(X,RIO)
0003      RIO=ABS(X)
0004      IF(RIO>3.75) 1,1,2
0005  1      Z=X*X*7.11111E-2
0006      RIO=((((4.5813E-3*Z+3.60768E-2)*Z+2.659732E-1)*Z+1.206749E0
0007      1089942E0)*Z+3.515623E0)*Z+1.
0008      RETURN
0009  2      Z=3.75/RIO
0010      RIO=EXP(RIO)/SQRT(RIO)*((((((3.92377E-3*Z-1.647633E-2)*Z+2
0011      17E-2)*Z-2.057706E-2)*Z+9.16281E-3)*Z-1.57565E-3)*Z+2.25319E-
0012      2+1.328592E-2)*Z+3.989423E-1)
0013      RETURN
0014      END
0015  END$
```

&BESJ T=00004 IS ON CRO0022 USING 00014 BLKS R=0129

```

0001 C
0002 C      .....
0003 C
0004 C      SUBROUTINE BESJ
0005 C
0006 C      PURPOSE
0007 C      COMPUTE THE J BESSLE FUNCTION FOR A GIVEN ARHUMENT AND
0008 C
0009 C      USAGE
0010 C      CALL BESJ(X,N,BJ,D,IER)
0011 C
0012 C      DESCRIPTION OF PARAMETERS
0013 C      X -THE ARGUMENT OF THE J BESSLE FUNCTION DESIRED
0014 C      N -THE ORDER OF THE J BESSLE FUNCTION DESIRED
0015 C      BJ -THE RESULTANT J BESSLE FUNCTION
0016 C      D -REQUIRED ACCURACY
0017 C      IER-RESULTANT ERROR CODE WHERE,
0018 C          IER=0 NO ERROR
0019 C          IER=1 N IS NEGATIVE
0020 C          IER=2 X IS NEGATIVE OR ZERO
0021 C          IER=3 REQUIRED ACCURACY NOT OBTAINED
0022 C          IER=4 RANGE OF N COMPARED TO X NOT CORRECT (SEE R
0023 C
0024 C      REMARKS
0025 C          N MUST BE GREATER THAN OR EQUAL TO ZERO, BUT IT MUST B
0026 C          LESS THAN
0027 C          20+10*X-X** 2/3    FOR X LESS THAN OR EQUAL TO 15
0028 C          90+X/2           FOR X GREATER THAN 15
0029 C
0030 C      SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
0031 C          NONE
0032 C
0033 C      METHOD
0034 C          RECURRENCE RELATION TECHNIQUE DESCRIBED BY H. GOLDSTEI
0035 C          R.M. THALER, 'RECURRENCE TECHNIQUES FOR THE CALCULATION
0036 C          BESSLE FUNCTIONS', M.T.A.C., V.13, PP.102-108 AND I.A. ST
0037 C          AND M. ABRAMOWITZ, 'GENERATION OF BESSLE FUNCTIONS ON H
0038 C          SPEED COMPUTERS', M.T.A.C., V.11, 1957, PP.255-257
0039 C
0040 C      .....
0041 C
0042 C      SUBROUTINE BESJ(X,N,BJ,D,IER)
0043 C
0044 C          BJ=.0
0045 C          IF(N)10,20,20
0046 10  C          IER=1
0047 C          RETURN
0048 20  C          IF(X)30,30,31
0049 30  C          IER=2
0050 C          RETURN
0051 31  C          IF(X-15.)32,32,34
0052 32  C          NTEST=20.+10.*X-(X**2/3))
0053 C          GO TO 36
0054 34  C          NTEST=90.+X/2.
0055 36  C          IF(N-NTEST)40,38,38
0056 38  C          IER=4
0057 C          RETURN

```

```

0058 40    IER=0
0059      N1=N+1
0060      BPREV=0.
0061 C
0062 C      COMPUTE STARTING VALUE OF M
0063 C
0064      IF(X-5.)50,60,60
0065 50      MA=X+6.
0066      GO TO 70
0067 60      MA=1.4*X+60./X
0068 70      MB=N+IFIX(X)/4+2
0069      MZERO=MAX0(MA,MB)
0070 C
0071 C      SET UPPER LIMIT OF M
0072 C
0073      MMAX=NTEST
0074 100     DO 190 M=MZERO,MMAX,3
0075 C
0076 C      SET F(M),F(M-1)
0077 C
0078      FM1=1.0E-28
0079      FM=.0
0080      ALPHA=.0
0081      IF(M-(M/2)*2)120,110,120
0082 110     JT=-1.
0083      GO TO 130
0084 120     JT=1.
0085 130     M2=M-2
0086      DO 160 K=1,M2
0087      MK=M-K
0088      BMK=2.*FLOAT(MK)*FM1/X-FM
0089      FM=FM1
0090      FM1=BMK
0091      IF(MK-N-1.)150,140,150
0092 140     BJ=BMK
0093 150     JT=-JT
0094     S=1+JT
0095 160     ALPHA=ALPHA+BMK*S
0096      BMK=2.*FM1/X-FM
0097      IF(N)180,170,180
0098 170     BJ=BMK
0099 180     ALPHA=ALPHA+BMK
0100     BJ=BJ/ALPHA
0101     IF(ABS(BJ-BPREV)-ABS(D*BJ))200,200,190
0102 190     BPREV=BJ
0103     IER=3
0104 200     RETURN
0105     END
0106  $
```

&BLDWF T=00004 IS ON CR00022 USING 00022 BLKS R=4113

```

0001  FTN4
0002      PROGRAM BLDWF
0003  C
0004  C
0005  C THIS PROGRAM IS USED IN CONJUNCTION WITH IMAGE PROCESSING
0006  C IT CREATES AND MAINTAINS AN IMAGE WORK FILE WITH PIXEL VALUES
0007  C STORED AS REAL NUMBERS TO PRESERVE PRECISION.
0008  C
0009  C
0010  C
0011  C
0012      DIMENSION IDCBL(272),IDCB2(1040),IDCB3(528),IMAGE(6),LU(5)
0013      DIMENSION IRTN(5),JNAME(3),IBUF(15),RDATA(512),IDATA(512),
0014      1 ISIZE(2)
0015  C
0016      EQUIVALENCE (ILINE,IRTN(4)),(IPIXL,IRTN(5)),(ILINE,RDATA),
0017      1(RDATA(2),RMAX),(RDATA(3),RMIN),(IBUF(12),ILOC),(IBUF(13),JN
0018      EQUIVALENCE (IBUF(7),NLINE),(IBUF(8),NPIXL),(IBUF(9),IPMIN),
0019      1 (IBUF(10),IPMAX),(ISIZE(2),ISIZ2)
0020  C
0021  C
0022  C
0023  C
0024  C GET INPUT PARAMETERS
0025  C
0026      CALL RMPAR(LU)
0027      IF (LU .LE. 0) LU = 1
0028  C
0029  C REUSE WORK FILE
0030  C
0031      WRITE(LU,7)
0032  7      FORMAT(//,"DO YOU WANT TO REUSE THE CURRENT WORK FILE? Y OR
0033      READ(LU,2) LANS
0034      IF(LANS .EQ. 1HY )GO TO 200
0035  C GET IMAGE NAME FROM USER
0036  C
0037      WRITE(LU,1)
0038  1      FORMAT("ENTER IMAGE NAME (12 CHARACTER)?_")
0039      READ(LU,2) IMAGE
0040  2      FORMAT(6A2)
0041  C
0042  C CHECK IF WORK FILE WANTED
0043  C
0044      IF (ICMPW(IMAGE,12H           ,6) .EQ. 0) GO TO 140
0045  C
0046  C OPEN DIRECTORY FILE
0047  C
0048  90      CALL OPEN(IDCBL,IERR,6HIMDIRC,1,2HIM,23,272)
0049      IF (IERR .LT. 0) GO TO 9999
0050  C

```

```
0051 C FIND IMAGE
0052 C
0053 100 CALL READF(IDCB1,IERR,IBUF,15,LEN)
0054 IF (IERR .LT. 0) GO TO 9999
0055 IF (LEN .EQ. -1) GO TO 9990
0056 C
0057 IF (ICMPW(IMAGE,IBUF,6) .NE. 0) GO TO 100
0058 C
0059 C IMAGE FOUND
0060 C
0061 C
0062 IF (ILOC .NE. 1) GO TO 9980
0063 C
0064 C IMAGE IS ON DISC
0065 C
0066 C CREATE WORK FILE
0067 C
0068 CALL OPEN(IDCB2,IERR,6HWF0000)
0069 IF (IERR .EQ. -6) GO TO 110
0070 IF (IERR .LT. 0) GO TO 9999
0071 C
0072 C ASK IF USER WANTS TO SAVE WORK FILE
0073 C
0074 WRITE(LU,6)
0075 6 FORMAT(" DO YOU WANT TO SAVE IMAGE IN CURRENT WORK FILE?_")
0076 READ(LU,2) IANS
0077 IF (IANS .EQ. 2HNO) GO TO 110
0078 C
0079 C SCHEDULE BUILD IMAGE PROGRAM
0080 C
0081 CALL CLOSE(IDCB2)
0082 CALL EXEC(23,6HBLDIM ,LU)
0083 C
0084 110 CALL PURGE(IDCB2,IERR,6HWF0000)
0085 IF (NPIXL .LT. 3) NPIXL = 3
0086 ISIZE = (2.0*FLOAT(NLINE+1)*FLOAT(NPIXL)+127.)/128.
0087 ISIZ2 = 2*NPIXL
0088 CALL CREAT(IDCB2,IERR,6HWF0000,ISIZE,2,0,0,1040)
0089 IF (IERR .LT. 0) GO TO 9999
0090 C
0091 C OPEN IMAGE DATA FILE
0092 C
0093 CALL OPEN(IDCB3,IERR,JNAME,1,2HIM,23,528)
0094 IF (IERR .LT. 0) GO TO 9999
0095 C
0096 C COPY DATA AND CONVERT TO REAL
0097 C
0098 C POSITION TO RECORD # 2
0099 C
0100 CALL WRITF(IDCB2,IERR,RDATA,1)
0101 C
0102 DO 120 I=1,NLINE
0103 CALL READF(IDCB3,IERR,IData,512,LEN)
0104 IF (IERR .LT. 0) GO TO 9999
0105 C
0106 DO 115 J=1,NPIXL
0107 115 RDATA(J) = IData(J)
0108 C
0109 CALL WRITF(IDCB2,IERR,RDATA)
```

```
0110      IF (IERR .LT. 0) GO TO 9999
0111 120  CONTINUE
0112 C
0113      RPMAX = IPMAX
0114      RPMIN = IPMIN
0115 C
0116 C  CLOSE ALL IMAGE FILES
0117 C
0118 130  CALL CLOSE(IDCBL)
0119      CALL CLOSE(IDCBL3)
0120 C
0121 C  WRITE INFO IN WORK FILE RECORD 1
0122 C
0123      ILINE = NLINE
0124      IPIXL = NPIXL
0125      RMAX = RPMAX
0126      RMIN = RPMIN
0127      CALL WRITF(IDCBL,IERR,RDATA,6,1)
0128      IF (IERR .LT. 0) GO TO 9999
0129 C
0130      CALL CLOSE(IDCBL2)
0131 C
0132 140  IRTN = 0
0133 200  CALL PRTN(IRTN)
0134      CALL EXEC(6)
0135 C
0136 C  ERRORS
0137 C
0138 C
0139 C  IMAGE NOT ON DISC
0140 C
0141 9980  WRITE(LU,4)
0142 4      FORMAT(" IMAGE NOT ON DISC!")
0143      IRTN = -100
0144      GO TO 200
0145 C
0146 C  IMAGE NOT FOUND
0147 C
0148 9990  WRITE(LU,3)
0149 3      FORMAT(" IMAGE NOT FOUND!")
0150      IRTN = -101
0151      GO TO 200
0152 C
0153 C  FILE ERROR
0154 C
0155 9999  WRITE(LU,5) IERR
0156 5      FORMAT("FILE ERROR =",I6)
0157 201  IF(IERR.EQ.-8) CALL CLOSE(IDCBL,IERR)
0158      IRTN = -103
0159      GO TO 200
0160      END
0161 $
```

```
0001  FTN4
0002      INTEGER FUNCTION SCROL(IDCB, IDIRC, NLINE, IFRST, ILAST, RMAX, RMIN)
0003  C
0004  C THIS SUBROUTINE IS USED TO SCROLL AN IMAGE ON THE GMR-27
0005  C
0006  C      IDCB = OPENED DATA CONTROL BLOCK FOR THE IMAGE
0007  C      IDIRC = DIRECTION TO SCROLL (-N= BACK N LINES N= FORWARD N LINES)
0008  C      NLINE = # LINES IN IMAGE
0009  C      IFRST = LOWEST IMAGE LINE DISPLAYED
0010  C      ILAST = HIGHEST IMAGE LINE DISPLAYED
0011  C
0012  C
0013      DIMENSION IDCB(144), IDATA(512)
0014  C
0015      INTEGER SCROL
0016  C
0017      DATA IUP, IDOWN / 34060B, 34040B /
0018  C
0019  C      CHECK IF NO WORK NECESSARY
0020  C
0021      IF (IDIRC .EQ. 0) RETURN
0022  C
0023      IF (IDIRC .GT. 0) GO TO 200
0024  C
0025  C      SCROLL IMAGE UP
0026  C
0027      DO 100 I=-1, IDIRC, -1
0028      IF (IFRST .LE. 0) RETURN
0029      CALL READF(IDCB, SCROL, IDATA, 512, LEN, IFRST)
0030  C
0031      DO 110 J=1, LEN
0032      IDATA(J) = (255. / (RMAX - RMIN)) * (IDATA(J) - RMIN)
0033      IF (IDATA(J) .LT. 0) IDATA(J) = 0
0034      IF (IDATA(J) .GT. 255) IDATA(J) = 255
0035  110      CONTINUE
0036  C
0037      IF (SCROL .LT. 0) RETURN
0038      CALL DRIVR(2, IUP, 1)
0039      CALL WLINE(0, 0, LEN-1, IDATA)
0040      IFRST = IFRST-1
0041  100      ILAST = ILAST-1
0042      RETURN
0043  C
0044  C      SCROLL IMAGE DOWN
0045  C
0046  200      DO 210 I=1, IDIRC
0047      IF (ILAST .GE. NLINE-1) RETURN
0048      CALL READF(IDCB, SCROL, IDATA, 512, LEN, ILAST+1)
0049  C
0050      DO 220 J=1, LEN
0051      IDATA(J) = (255. / (RMAX - RMIN)) * (IDATA(J) - RMIN)
0052      IF (IDATA(J) .LT. 0) IDATA(J) = 0
0053      IF (IDATA(J) .GT. 255) IDATA(J) = 255
0054  220      CONTINUE
0055  C
0056      IF (SCROL .LT. 0) RETURN
0057      CALL DRIVR(2, IDOWN, 1)
0058      CALL WLINE(255, 0, LEN-1, IDATA)
0059      ILAST = ILAST+1
0060  210      IFRST = IFRST+1
0061  C
0062      RETURN
0063      END
```

SWLINE T=00004 IS ON CRO0022 USING 00005 BLKS R=0036

```
0001  FTN4,L
0002      SUBROUTINE WLINE(LINE,IPIX,JPIX,IData)
0003  C
0004  C  THIS SUBROUTINE WRITES A DESIGNATED LINE TO THE GMR-27
0005  C
0006  C      LINE = LINE NUMBER
0007  C      IPIX = STARTING PIXEL
0008  C      JPIX = ENDING PIXEL
0009  C      IData = BUFFER CONTAINING IMAGE DATA FOR LINE
0010  C
0011  C
0012      DIMENSION IData(512),INIT(6)
0013  C
0014      EQUIVALENCE (LLA,INIT(2)),(LEA,INIT(3)),(LEB,INIT(4))
0015  C
0016      DATA INIT/100377B,64000B,44000B,50000B,24041B,26002B/
0017  C
0018  C  COMPUTE DIRECTION
0019  C
0020      IDIRC = 1
0021      IF (IPIX .GT. JPIX) IDIRC = -1
0022  C
0023  C  SET UP TO WRITE LINE
0024  C
0025      LLA = 64000B + IAND(LINE,377B)
0026      LEA = 44000B + IAND(IPIX,777B)
0027      LEB = 50000B + IDIRC + 512
0028      CALL DRIVR(2,INIT,6)
0029  C
0030  C  WRITE LINE
0031  C
0032      NUM = IDIRC*(JPIX-IPIX)+1
0033      CALL DRIVR(2,IData,NUM)
0034  C
0035      RETURN
0036
0037  $
```

&DRIVR T=00004 IS ON CR00022 USING 00012 BLKS R=0241

```

0001 ASMB,R,L,C
0002     NAM DRIVR,6
0003     ENT DRIVR
0004     EXT .ENTR,$LIBR,$LIBX
0005 *
0006 *
0007 OPCOD BSS 1
0008 BUFR  BSS 1
0009 LEN   BSS 1
0010 *
0011 DRIVR NOP      ENTRY
0012     JSB .ENTR   GET
0013     DEF OPCOD  PARAMETERS.
0014     LDA LEN,I   GET # WORDS
0015     CMA,INA   NEGATE
0016     STA CNT   & SAVE.
0017     SSA,RSS   IF NOT NEGATIVE
0018     JMP EXIT   EXIT
0019 *
0020     JSB $LIBR  TURN OFF
0021     NOP      INTERRUPTS.
0022     LDA OPCOD,I CHECK REQUEST
0023     SLA,ELA   IF READ
0024     JMP D.2   GO PROCESS
0025 *
0026 *  WRITE REQUEST
0027 *
0028     SSA,RSS   IF DMA NOT REQUIRED
0029     JMP D.1   GO DO PROGRAMMED I O
0030 *
0031 *  DMA OUTPUT
0032 *
0033     LDA CW1   GET CONTROL WORD 1
0034     OTA DMA2  USE CHANNEL 2
0035     CLC 3B   PREPARE TO SEND ADDRESS
0036     LDA BUFR
0037     OTA 3B
0038     STC 3B   PREPARE TO SEND COUNT
0039     LDA CNT
0040     OTA 3B
0041     LDA BUFR,I
0042     OTA SC
0043     STC SC,C   START DEVICE
0044     STC DMA2,C START DMA
0045     SFS DMA2
0046     JMP *-1
0047     CLF DMA2
0048     JMP EXIT+1
0049 *
0050 *

```

```

0051 D.1  LDA BUFR,I  GET DATA WORD
0052     OTA SC      OUTPUT IT.
0053     STC SC,C    TURN ON DEVICE
0054     SFS SC      WAIT 'TIL
0055     JMP *-1     DONE
0056     ISZ BUFR    BUMP BUFFER ADDRESS
0057     ISZ CNT     LAST WORD?
0058     JMP D.1     NO GO BACK.
0059     JMP EXIT    GO EXIT
0060 *
0061 *  READ ENTRY
0062 *
0063 D.2  SSA          SKIP IF SPECIAL
0064     JMP D.3     MODE
0065     LDA SPD8    SET UP
0066     OTA SC
0067     STC SC,C    FOR
0068     SFS SC
0069     JMP *-1     READ.
0070 D.3  LDA RDPD    GET READ DATA CODE
0071     OTA SC
0072     STC SC,C    START DEVICE
0073     SFS SC      WAIT 'TIL
0074     JMP *-1
0075 D.4  LDA RDPD
0076     OTA SC
0077     STC SC,C
0078     SFS SC
0079     JMP *-1
0080     LIA SC      DONE.  GET WORD.
0081     STA BUFR,I  STUFF IN BUFFER
0082     ISZ BUFR    BUMP BUFFER
0083     ISZ CNT     DONE?
0084     JMP D.4     NO GO BACK.
0085 *
0086 EXIT  CLC SC    TURN OFF DEVICE
0087     JSB $LIBX   RESTORE RTE AND
0088     DEF DRIVR  RETURN
0089 *
0090 *
0091 *
0092 A    EQU 0
0093 *
0094 SC   EQU 22B
0095 RDPD OCT 160000
0096 SPD8 OCT 120400
0097 CNT  BSS 1
0098 CW1  OCT 120022  * HAVE TO CHANGE WITH SELECT CODE
0099 DMA2 EQU 7
0100 END

```

&FDIG1 T=00004 IS ON CR00022 USING 00018 BLKS R=0132

```

0001  FTN4,L
0002      SUBROUTINE ROTAEC(U,V,MN,LU)
0003      COMPLEX P(10),Q(10),QQ,PP
0004      DIMENSION U(3,3,2),V(3,3,2)
0005      COMMON/WORK/AMAG(10),A(3,3),B(3,3)
0006      WRITE(LU,100)
0007  100  FORMAT(" SELECT FILTER /*, 1. BUTTERWORTH /*,
0008      1  " 2. CHEBYSHEV /*, 3. LINEAR PHASE /*)
0009      READ(LU,*) ITYPE
0010      WRITE(LU,110)
0011  110  FORMAT(" ENTER THE NUMBER OF FILTER STAGES /*)
0012      READ(LU,*) NSTG
0013      WRITE(LU,120)
0014  120  FORMAT(" ENTER RELATIVE CUTOFF FREQUENCY FOR LOWPASS /*)
0015      READ(LU,*) WR
0016      WRITE(LU,140)
0017  140  FORMAT(" ARE ALL ZEROS LOCATED AT INFINITY /*,
0018      1  " 1 = YES /*, 2 = NO /*)
0019      READ(LU,*) IFLAG
0020      WRITE(LU,151)
0021  151  FORMAT(" ENTER RIPPLE FACTOR /*)
0022      READ(LU,*) ELP
0023  C
0024  C      IF(ITYPE.EQ.1) CALL BUTTER
0025  C      IF(ITYPE.EQ.2) CALL CHEB1(NSTG,WR,P,AMAG,ELP)
0026  C      IF(ITYPE.EQ.3) CALL LINEAR PHASE
0027  C
0028  20   DO 10 J=1,NSTG
0029  30   WRITE(LU,130) J
0030  130  FORMAT(" ENTER ROTATION ANGLE IN NEG. DEGREES FOR STAGE #*
0031  1,I2/)
0032      READ(LU,*) THETA
0033  C
0034      PMAG = AMAG(J)
0035      Q(J) = CMPLX(-1.,0.)
0036      QQ = Q(J)
0037      PP = P(J)
0038      CALL SROTT(A,B,PMAG,PP,QQ,IFLAG,THETA)
0039      DO 1111 I=1,3
0040      DO 1111 K=1,3
0041      U(I,K,J) = A(I,K)
0042  1111  V(I,K,J) = B(I,K)
0043      WRITE(LU,40) P(J),AMAG(J)
0044  40   FORMAT(1X,1(" P=",1E15.5," +J",1E15.5,/)," PMAG= ",E15.5
0045  10   CONTINUE
0046      MN = NSTG + 1
0047      WRITE(1,1112) U
0048      WRITE(1,1112) V
0049  1112  FORMAT(3E15.4)
0050      RETURN
0051      END

```

```
0052      SUBROUTINE CHEB1(N,WR,P,AMAG,ELP)
0053      DIMENSION AMAG(N)
0054      COMPLEX P(N),PN
0055      PI=3.1415927
0056      E=1.0/ELP
0057      SINHIV=ALOG(E+SQRT(E**2+1.0))
0058      ALP=(-1.0*SINHIV)/FLOAT(N)
0059      IF(WR.EQ.1.0) GO TO 30
0060      X=0.5*WR*PI
0061      IF(COS(X).EQ.0.0) GOTO 30
0062      XTAN=SIN(X)/COS(X)
0063      KK=1
0064      NTWO=4*N
0065      XX=1.0/FLOAT(NTWO)
0066      DO 20 I=1,NTWO
0067      GAMMA=(2*I-1)*PI*XX
0068      C1=(EXP( ALP)-EXP(-ALP))/2.
0069      C2=SIN(GAMMA)
0070      C3=(EXP( ALP)+EXP(-ALP))/2.
0071      C4=COS(GAMMA)
0072      XR=C1*C2
0073      XI=C3*C4
0074      PN=XTAN*CMPLX(XR,XI)
0075      IF(REAL(PN).GT.0.0) GO TO 20
0076      IF(AIMAG(PN).LT.0.0) GO TO 20
0077      P(KK)=PN
0078      AMAG(KK)=CABS(PN)**2
0079      20   KK=KK+1
0080      GO TO 34
0081      30   WRITE(LU,33)
0082      33   FORMAT("    CUTOFF FREQ. CAN NOT = 1.0 ")
0083      34   RETURN
0084      END
```

```

0085      SUBROUTINE SROTT(A,B,PMAG,PP,QQ,IFLAG,THETA)
0086      DIMENSION A(3,3),B(3,3)
0087      COMPLEX PP,QQ
0088      ADJ=0.999
0089      X=THETA*0.0174533
0090      C1=COS(X)**2
0091      C2=-2.0*COS(X)*SIN(X)
0092      C3=SIN(X)**2
0093      C7=-2.0*REAL(PP)*COS(X)
0094      C8=2.0*REAL(PP)*SIN(X)
0095      C9=CABS(PP)**2
0096      B(1,1)=C1+C2+C3+C7+C8+C9
0097      B(1,2)=2.0*(C1-C3+C7+C9)*ADJ
0098      B(1,3)=(C1-C2+C3+C7-C8+C9)*ADJ**2
0099      B(2,1)=2.0*(C3-C1+C8+C9)*ADJ
0100      B(2,2)=4.0*(C9-C1-C3)*ADJ**2
0101      B(2,3)=2.0*(C3-C1-C8+C9)*ADJ**3
0102      B(3,1)=(C1-C2+C3-C7+C8+C9)*ADJ**2
0103      B(3,2)=2.0*(C1-C3-C7+C9)*ADJ**3
0104      B(3,3)=(C1+C2+C3-C7-C8+C9)*ADJ**4
0105      IF(IFLAG.EQ.1) GO TO 10
0106      C4=-2.0*REAL(QQ)*COS(X)
0107      C5=2.0*REAL(QQ)*SIN(X)
0108      C6=CABS(QQ)**2
0109      A(1,1)=C1+C2+C3+C4+C5+C6
0110      A(1,2)=2.0*(C1-C3+C4+C6)
0111      A(1,3)=C1-C2+C3+C4-C5+C6
0112      A(2,1)=2.0*(C3-C1+C5+C6)
0113      A(2,2)=4.0*(C6-C1-C3)
0114      A(2,3)=2.0*(C3-C1-C5+C6)
0115      A(3,1)=C1-C2+C3-C4+C5+C6
0116      A(3,2)=2.0*(C1-C3-C4+C5+C6)
0117      A(3,3)=C1+C2+C3-C4-C5+C6
0118      GO TO 20
0119 10      A(1,1)=1.0
0120      A(1,2)=2.0
0121      A(1,3)=1.0
0122      A(2,1)=2.0
0123      A(2,2)=4.0
0124      A(2,3)=2.0
0125      A(3,1)=1.0
0126      A(3,2)=2.0
0127      A(3,3)=1.0
0128 20      CONTINUE
0129      SCAL = 1./B(1,1)
0130      DO 30 I=1,3
0131      DO 30 K=1,3
0132          B(I,K)=( B(I,K)*SCAL)
0133          A(I,K)=(A(I,K)*SCAL*PMAG)
0134 30      CONTINUE
0135          RETURN
0136          END
0137  $          $
0138  $          $

```

&STABI T=00004 IS ON CRO0022 USING 00070 BLKS R=0668

```

0001  FTN4,L
0002      PROGRAM START
0003  C
0004  C THIS PROGRAM EVALUATES THE FILTER STABILITY CHARACTERISTICS
0005  C
0006  C
0007      COMMON/WORK/WO(130)
0008  C      INTEGER BUFF
0009      DIMENSION IBUF(80),ILU(5),IRTN(5)
0010      DIMENSION V(3,3,2),U(3,3,2)
0011      EQUIVALENCE (IBUF(1),U(1,1,1)),(IBUF(41),V(1,1,1))
0012  C
0013      CALL RMPAR(ILU)
0014      LU=ILU(1)
0015      MN=ILU(2) + 1
0016  C
0017  C
0018  C GET FILTER COEFF'S
0019      CALL EXEC(14,1,IBUF,80)
0020  C
0021  C
0022      CALL STABT(V,MN,IRTCD,LU)
0023      IRTN = IRTCD
0024  C
0025      CALL PRTN(IRTN)
0026      END
0027      SUBROUTINE STABT(V,MN,IRTCD,LU)
0028  C      SUBROUTINE CHECKS STABILITY OF SYSTEM EQUATION-
0029  C      Y(M,N)=A*Y(M-1,N)+B*Y(M,N-1)
0030  C
0031  C      C---COEFFICIENT MATRIX OF DENOMINATON OF ZW-TRANSFORM OF SYS
0032  C      IMPULSE FUNCTION
0033  C
0034      LOGICAL ISTAB
0035      DIMENSION V(3,3,2)
0036      DIMENSION C(5,5),A(25,25),B(25,25),S(25,25),EVR(25),EVI(25)
0037      COMMON/WORK/IERR(25)
0038      MDIM=25
0039      N=2*(MN-1)+1
0040      M=N**2
0041      IF(MN.EQ.3) GO TO 5
0042  C
0043  C      PUT COEFFICIENTS IN STABILITY ARRAY
0044  C
0045      DO 6 I=1,3
0046      DO 6 J=1,3
0047      6 C(I,J)=V(I,J,1)
0048      GO TO 13
0049      5 DO 10 I=1,5
0050      DO 10 J=1,5
0051      DO 10 K=1,3
0052      DO 10 L=1,3
0053      IK=I-K+1
0054      JL=J-L+1
0055      IF((IK .LE. 0) .OR. (IK .GT. 3)) GO TO 10
0056      IF((JL .LE. 0) .OR. (JL .GT. 3)) GO TO 10
0057      C(I,J)=C(I,J)+V(IK,JL,1)*V(K,L,2)
0058      10 CONTINUE

```

```

0059 C
0060 13 CONTINUE
0061 C   WRITE(LU,11)
0062 11 FORMAT(20HO COEFFICIENT MATRIX,/ )
0063 C   DO 21 I=1,N
0064 C   21 WRITE(LU,12) (C(I,J),J=1,N),N
0065 12 FORMAT(1H ,5F15.6)
0066 C
0067 C   FORM A AND B MATRICES
0068 C
0069   DO 22 I=1,M
0070   DO 22 J=1,M
0071   A(I,J)=0.0
0072   B(I,J)=0.0
0073 22 S(I,J)=0.0
0074   NNOW=N-1
0075   DO 23 J=1,N
0076   DO 23 I=1,NNOW
0077   K=I+(J-1)*N
0078   IF(J.EQ.1) GO TO 24
0079 24 A(1,K)=-C(I+1,J)
0080   IF(J.GT.1) A(1,K)=-0.5*C(I+1,J)
0081   IF(J.GT.1) A(K+1,K)=0.5
0082   IF(J.EQ.1) A(K+1,K)=1.0
0083 23 CONTINUE
0084   DO 25 J=1,NNOW
0085   DO 25 I=1,N
0086   K=I+(J-1)*N
0087   KN=K+N
0088   IF(I.EQ.1) GO TO 26
0089 26 B(1,K)=-C(I,J+1)
0090   IF(I.GT.1) B(1,K)=-0.5*C(I,J+1)
0091   IF(I.GT.1) B(KN,K)=0.5
0092   IF(I.EQ.1) B(KN,K)=1.0
0093 25 CONTINUE
0094 C
0095 C
0096 C   FIND EIGENVALUES OF A AND B
0097 C
0098   DO 27 I=1,M
0099   DO 27 J=1,M
0100 27 S(I,J)=A(I,J)
0101   CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
0102   WRITE(LU,71)
0103 71 FORMAT(/,10X,19HEIGEN VALUES OF (A))
0104 C
0105   TEST=1.0
0106   IONE=0
0107 C
0108   CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0109   IF(ISTAB) GOTO 405
0110 400 FORMAT(" FILTER IS UNSTABLE!/")
0111 401 FORMAT(" FILTER IS STABLE/")
0112 C
0113   DO 94 I=1,M
0114   DO 94 J=1,M
0115 94 S(I,J)=0.0
0116   DO 28 I=1,M
0117   DO 28 J=1,M
0118 28 S(I,J)=B(I,J)

```

```

0119      CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
0120      WRITE(LU,72)
0121      72 FORMAT(/,10X,19HEIGEN VALUES OF (B))
0122      CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0123      IF(ISTAB) GOTO 405
0124      C
0125      C      FIND EIGENVALUES OF A+B
0126      C
0127      DO 29 I=1,M
0128      DO 29 J=1,M
0129      29 S(I,J)=A(I,J)+B(I,J)
0130
0131      CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
0132      WRITE(LU,73)
0133      73 FORMAT(/,10X,21HEIGEN VALUES OF (A+B))
0134      CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0135      405 IF(ISTAB) WRITE(LU,400)
0136      IF(ISTAB) GO TO 404
0137      WRITE(LU,401)
0138      404  IRTCD = 0
0139      GO TO 500
0140      C
0141      C      FIND EIGENVALUES OF A*S
0142      C
0143      DO 30 I=1,M
0144      DO 30 J=1,M
0145      30 S(I,J)=0.0
0146      DO 31 I=1,N
0147      DO 31 J=1,N
0148      K=J+(I-1)*N
0149      L=I+(J-1)*N
0150      31 S(K,L)=1.0
0151      CALL MLTMX(A,S,M,MDIM)
0152      CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
0153      WRITE(LU,74)
0154      74 FORMAT(/,10X,21HEIGEN VALUES OF (A*S))
0155      C
0156      IONE=1
0157      TEST=0.5
0158      C
0159      ICNT=0
0160      CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0161      IF(ISTAB) ICNT=1
0162      DO 230 I=1,M
0163      DO 230 J=1,M
0164      230 S(I,J)=0.0
0165      DO 231 I=1,N
0166      DO 231 J=1,N
0167      K=J+(I-1)*N
0168      L=I+(J-1)*N
0169      231 S(K,L)=1.0
0170      CALL MLTMX(B,S,M,MDIM)
0171      CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
0172      WRITE(LU,75)
0173      75 FORMAT(/,10X,21HEIGEN VALUES OF (B*S))
0174      CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0175      IF(ISTAB) ICNT=ICNT+1
0176      IF(ICNT.EQ.2) WRITE(LU,401)

```

```

0177 C
0178 C      FIND EIGENVALUES OF ABS(A)+ABS(B)
0179 C
0180      DO 33 I=1,M
0181      DO 33 J=1,M
0182      33 S(I,J)=ABS(A(I,J))+ABS(B(I,J))
0183      CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
0184      WRITE(LU,76)
0185      76 FORMAT(/,10X,29HEIGEN VALUES OF ABS(A)+ABS(B))
0186 C
0187      TEST=1.0
0188 C
0189      CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0190      IF(ISTAB) WRITE(LU,401)
0191 C
0192 C      FIND EIGENVALUES OF A*B
0193 C
0194      CALL MLTMX(A,B,M,MDIM)
0195      CALL RNAN(MDIM,M,S,EVR,EVI,IERR)
0196      WRITE(LU,77)
0197      77 FORMAT(/,10X,21HEIGEN VALUES OF (A*B))
0198      CALL PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0199      IF(ISTAB) WRITE(LU,401)
0200      GO TO 501
0201 500      IF(ISTAB) IRTCD = 1000
0202 501      RETURN
0203      END
0204      SUBROUTINE PNTEV(EVR,EVI,M,MDIM,TEST,IONE,ISTAB,IERR,LU)
0205      LOGICAL ISTAB
0206      DIMENSION EVR(MDIM),EVI(MDIM),IERR(MDIM)
0207 C
0208      ISTAB=.FALSE.
0209      D=1.0E-20
0210      RMX=0.0
0211      DO 20 I=1,M
0212      R=EVR(I)**2+EVI(I)**2
0213      R=SQRT(R)
0214      RMX=AMAX1(RMX,R)
0215      IF(R.LT.D) GO TO 20
0216      IF(IERR(I).LT.0) WRITE(LU,93) I,IERR(I)
0217 20      CONTINUE
0218      WRITE(LU,30) RMX
0219 C
0220      IF(IONE.EQ.0.AND.RMX.GE.TEST) ISTAB=.TRUE.
0221      IF(IONE.EQ.1.AND.RMX.LE.TEST) ISTAB=.TRUE.
0222 10      FORMAT(1H ,E14.7,4X,2H+J,E14.7)
0223 11      FORMAT(13H ABS(LMDA) = ,E14.7)
0224 30      FORMAT(19H SPECTRAL RADIUS = ,E14.7//)
0225 93      FORMAT(//,10X,"IERR(",I2,") = ",I2/)
0226      RETURN
0227      END
0228      SUBROUTINE RNAN(N,M,S,EVR,EVI,IERR)
0229 C      SUBROUTINE WAS WRITTEN TO CALL HSBG AND ATEIG IBM SCIENTIFIC
0230 C      SUBROUTINES TO CALCULATE THE EIGENVALUES OF A REAL MATR
0231 C      M----ORDER OF THE MATRIX S
0232 C      N----SIZE OF FIRST DIMENSION ASSIGNED TO THE ARRAY S IN THE
0233 C      CALLING PROGRAM

```

```

0234 C
0235      DIMENSION S(25,25),EVR(25),EVI(25)
0236      COMMON/WORK/LANA(25)
0237      CALL HSBG(M,S,N)
0238      CALL ATEIG(M,S,EVR,EVI,LANA,N)
0239      RETURN
0240      END

0241 C      SUBROUTINE ATEIG
0242 C      PURPOSE
0243 C          COMPUTE THE EIGENVALUES OF A REAL ALMOST TRIANGULAR MA
0244 C
0245 C      USAGE
0246 C          CALL ATEIG(M,A,RR,RI,LANA,IA)

0248 C      DESCRIPTION OF THE PARAMETERS
0249 C          M      ORDER OF THE MATRIX
0250 C          A      THE INPUT MATRIX, M BY M
0251 C          RR     VECTOR CONTAINING THE REAL PARTS OF THE EIGENVA
0252 C          ON RETURN
0253 C          RI     VECTOR CONTAINING THE IMAGINARY PARTS OF THE EI
0254 C          VALUES ON RETURN
0255 C          LANA    VECTOR WHOSE DIMENSION MUST BE GREATER THAN OR
0256 C          TO M, CONTAINING ON RETURN INDICATIONS ABOUT TH
0257 C          THE EIGENVALUES APPEARED (SEE MATH. DESCRIPTION
0258 C          IA      SIZE OF THE FIRST DIMENSION ASSIGNED TO THE ARR
0259 C          IN THE CALLING PROGRAM WHEN THE MATRIX IS IN DO
0260 C          SUBSCRIPTED DATA STORAGE MODE.
0261 C          IA=M WHEN THE MATRIX IS IN SSP VECTOR STORAGE M
0262 C
0263 C      REMARKS
0264 C          THE ORIGINAL MATRIX IS DESTROYED
0265 C          THE DIMENSION OF RR AND RI MUST BE GREATER OR EQUAL TO
0266 C
0267 C      SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
0268 C          NONE
0269 C
0270 C      METHOD
0271 C          QR DOUBLE ITERATION
0272 C
0273 C      REFERENCES
0274 C          J.G.F. FRANCIS - THE QR TRANSFORMATION---THE COMPUTER
0275 C          JOURNAL, VOL. 4, NO. 3, OCTOBER 1961, VOL. 4, NO. 4, J
0276 C          1962. J. H. WILKINSON - THE ALGEBRAIC EIGENVALUE PROB
0277 C          CLarendon Press, Oxford, 1965.
0278 C
0279 C          .....
0280 C
0281      SUBROUTINE ATEIG(M,A,RR,RI,LANA,IA)
0282      DIMENSION A(1),RR(1),RI(1),PRR(2),PRI(2),LANA(1)
0283      INTEGER P,P1,Q
0284 C
0285      E7=1.0E-8
0286      E6=1.0E-6
0287      E10=1.0E-10
0288      DELTA=0.5
0289      MAXIT=30

```

```

0290 C
0291 C      INITIALIZATION
0292 C
0293      N=M
0294 20 N1=N-1
0295      IN=N1*IA
0296      NN=IN+N
0297      IF(N1) 30,1300,30
0298 30 NP=N+1
0299 C
0300 C      ITERATION COUNTER
0301 C
0302      IT=0
0303 C
0304 C      ROOTS OF THE 2ND ORDER MAIN SUBMATRIX AT THE PREVIOUS
0305 C      ITERATION
0306 C
0307 DO 40 I=1,2
0308      PRR(I)=0.0
0309 40 PRI(I)=0.0
0310 C
0311 C      LAST TWO SUBDIAGONAL ELEMENTS AT THE PREVIOUS ITERATION
0312 C
0313      PAN=0.0
0314      PAN1=0.0
0315 C
0316 C      ORIGIN SHIFT
0317 C
0318      R=0.0
0319      S=0.0
0320 C
0321 C      ROOTS OF THE LOWER MAIN 2 BY 2 SUBMATRIX
0322 C
0323      N2=N1-1
0324      V1=IN-IA
0325      NN1=IN1+N
0326      N1N=IN+N1
0327      N1N1=IN1+N1
0328 60 T=A(N1N1)-A(NN)
0329      U=T*T
0330      V=4.0*A(N1N)*A(NN1)
0331      IF(ABS(V)-U*E7) 100,100,65
0332 65 T=U+V
0333      IF(ABS(T)-AMAX1(U,ABS(V))*E6) 67,67,68
0334 67 T=0.0
0335 68 U=(A(N1N1)+A(NN))/2.0
0336      V=SQRT(ABS(T))/2.0
0337      IF(T)140,70,70
0338 70 IF(U) 80,75,75
0339 75 RR(N1)=U+V
0340      RR(N)=U-V
0341      GO TO 130
0342 80 RR(N1)=U-V
0343      RR(N)=U+V
0344      GO TO 130
0345 100 IF(T)120,110,110
0346 110 RR(N1)=A(N1N1)
0347      RR(N)=A(NN)
0348      GO TO 130

```

```

0349 120 RR(N1)=A(NN)
0350  RR(N)=A(N1N1)
0351 130 RI(N)=0.0
0352  RI(N1)=0.0
0353  GO TO 160
0354 140   RR(N1) = U
0355   RR(N)=U
0356   RI(N1)=V
0357   RI(N)=-V
0358 160 IF(N2)1280,1280,180
0359 C
0360 C      TESTS OF CONVERGENCE
0361 C
0362 180 N1N2=N1N1-IA
0363   RMOD=RR(N1)*RR(N1)+RI(N1)*RI(N1)
0364   EPS=E10*SQRT(RMOD)
0365   IF(ABS(A(N1N2))-EPS)1280,1280,240
0366 240 IF(ABS(A(NN1))-E10*ABS(A(NN))) 1300,1300,250
0367 250 IF(ABS(PAN1-A(N1N2))-ABS(A(N1N2))*E6) 1240,1240,260
0368 260 IF(ABS(PAN-A(NN1))-ABS(A(NN1))*E6)1240,1240,300
0369 300 IF(IT-NAXIT) 320,1240,1240
0370 C
0371 C      COMPUTE THE SHIFT
0372 C
0373 320 J=1
0374   DO 360 I = 1,2
0375   K=NP-I
0376   IF(ABS(RR(K)-PRR(I))+ABS(RI(K)-PRI(I))-DELTA*(ABS(RR(K))
0377   1      *S(RI(K)))) 340,360,360
0378 340 J=J+1
0379 360 CONTINUE
0380   GO TO (440,460,460,480),J
0381 440 R=0.0
0382   S=0.0
0383   GO TO 500
0384 460 J=N+2-J
0385   R=RR(J)*RR(J)
0386   S=RR(J)+RR(J)
0387   GO TO 500
0388 480 R=RR(N)*RR(N1)-RI(N)*RI(N1)
0389   S=RR(N)+RR(N1)
0390 C
0391 C      SAVE THE LAST TWO SUBDIAGONAL TERMS AND THE ROOTS OF THE
0392 C      SUBMATRIX BEFORE ITERATION
0393 C
0394 500 PAN=A(NN1)
0395   PAN1=A(N1N2)
0396   DO 520 I=1,2
0397   K=NP-I
0398   PRR(I)=RR(K)
0399 520 PRI(I)=RI(K)
0400 C
0401 C      SEARCH FOR A PARTITION OF THE MATRIX, DEFINED BY P AND Q
0402 C
0403   P=N2
0404   IF (N-3)600,600,525
0405 525 IPI=N1N2
0406   DO 580 J=2,N2
0407   IPI=IPI-IA-1
0408   IF(ABS(A(IPI))-EPS) 600,600,530
0409 530 IPIP=IPI+IA

```

```

0410      IPIP2=IPIP+LA
0411      D=A(IPIP)*(A(IPIP)-S)+A(IPIP2)*A(IPIP+1)+R
0412      IF(D)540,560,540
0413 540 IF(ABS(A(IPI)*A(IPIP+1))*(ABS(A(IPIP)+A(IPIP2+1)-S)+ABS(A(IP
0414 1 ))-ABS(D)*EPS) 620,620,560
0415 560 P=N1-J
0416 580 CONTINUE
0417 600 Q=P
0418      GO TO 680
0419 620 P1=P-1
0420      Q=P1
0421      IF (P1-1) 680,680,650
0422 650 DO 660 I=2, P1
0423      IPI=IPI-IA-1
0424      IF(ABS(A(IPI))-EPS)680,680,660
0425 660 Q=Q-1
0426 C
0427 C      QR DOUBLE ITERATION
0428 C
0429 680 II=(P-1)*LA+P
0430      DO 1220 I=P,N1
0431      III=II-IA
0432      IIP=II+IA
0433      IF(I-P)720,700,720
0434 700 IPI=II+1
0435      IPIP=IIP+1
0436 C
0437 C      INITIALIZATION OF THE TRANSFORMATION
0438 C
0439      G1=A(II)*(A(II)-S)+A(IIP)*A(IPI)+R
0440      G2=A(IPI)*(A(IPIP)+A(II)-S)
0441      G3=A(IPI)*A(IPIP+1)
0442      A(IPI+1)=0.0
0443      GO TO 780
0444 720 G1=A(III)
0445      G2=A(III+1)
0446      IF(I-N2)740,740,760
0447 740 G3=A(III+2)
0448      GO TO 780
0449 760 G3=0.0
0450 780 CAP=SQRT(G1*G1+G2*G2+G3*G3)
0451      IF(CAP)800,860,800
0452 800 IF(G1)820,840,840
0453 820 CAP=-CAP
0454 840 T=G1+CAP
0455      PSI1=G2/T
0456      PSI2=G3/T
0457      ALPHA=2.0/(1.0+PSI1*PSI1+PSI2*PSI2)
0458      GO TO 880
0459 860 ALPHA=2.0
0460      PSI1=0.0
0461      PSI2=0.0
0462 880 IF(I-Q)900,960,900
0463 900 IF(I-P)920,940,920
0464 920 A(III)=-CAP
0465      GO TO 960
0466 940 A(III)=-A(III)
0467 C
0468 C      ROW OPERATION
0469 C

```

```

0470  960 IJ=II
0471      DO 1040 J=I,N
0472      T=PSI1*A(IJ+1)
0473      IF(I-N1)980,1000,1000
0474  980 IP2J=IJ+2
0475      T=T+PSI2*A(IP2J)
0476  1000 ETA=ALPHA*(T+A(IJ))
0477      A(IJ)=A(IJ)-ETA
0478      A(IJ+1)=A(IJ+1)-PSI1*ETA
0479      IF(I-N1)1020,1040,1040
0480  1020 A(IP2J)=A(IP2J)-PSI2*ETA
0481  1040 IJ=IJ+IA
0482 C
0483 C      COLUMN OPERATION
0484 C
0485      IF(I-N1)1080,1060,1060
0486  1060 K=N
0487      GO TO 1100
0488  1080 K=I+2
0489  1100 IP=IIP-I
0490      DO 1180 J=Q,K
0491      JIP=IP+J
0492      JI=JIP-LA
0493      T=PSI1*A(JIP)
0494      IF(I-N1)1120,1140,1140
0495  1120 JIP2=JIP+IA
0496      T=T+PSI2*A(JIP2)
0497  1140 ETA=ALPHA*(T+A(JI))
0498      A(JI)=A(JI)-ETA
0499      A(JIP)=A(JIP)-ETA*PSI1
0500      IF(I-N1)1160,1180,1180
0501  1160 A(JIP2)=A(JIP2)-ETA*PSI2
0502  1180 CONTINUE
0503      IF(I-N2)1200,1220,1220
0504  1200 JI=II+3
0505      JIP=JI+IA
0506      JIP2=JIP+IA
0507      ETA=ALPHA*PSI2*A(JIP2)
0508      A(JI)=-ETA
0509      A(JIP)=-ETA*PSI1
0510      A(JIP2)=A(JIP2)-ETA*PSI2
0511  1220 II=IIP+1
0512      IT=IT+1
0513      GO TO 60
0514 C
0515 C      END OF ITERATION
0516 C
0517  1240 IF(ABS(A(NN1))-ABS(A(N1N2))) 1300,1280,1280
0518 C
0519 C      TWO EIGENVALUES HAVE BEEN FOUND
0520 C
0521  1280 LANA(N)=0
0522      LANA(N1)=2
0523      N=N2
0524      IF(N2)1400,1400,20
0525 C
0526 C      ONE EIGENVALUE HAS BEEN FOUND
0527 C

```

```

0528 1300 RR(N)=A(NN)
0529      RI(N)=0.0
0530      IANA(N)=1
0531      IF(N1)1400,1400,1320
0532 1320 N=N1
0533      GO TO 20
0534 1400 RETURN
0535      END
0536 C      SUBROUTINE HSBG
0537 C
0538 C      PURPOSE
0539 C      TO REDUCE A REAL MATRIX INTO UPPER ALMOST TRIANGULAR F
0540 C
0541 C      USAGE
0542 C      CALL HSBG(N,A,IA)
0543 C
0544 C      DESCRIPTION OF THE PARAMETERS
0545 C      N      ORDER OF THE MATRIX
0546 C      A      THE INPUT MATRIX, N BY N
0547 C      IA     SIZE OF THE FIRST DIMENSION ASSIGNED TO THE ARR
0548 C                  A IN THE CALLING PROGRAM WHEN THE MATRIX IS IN
0549 C      DOUBLE SUBSCRIPTED DATA STORAGE MODE.  IA=N WHE
0550 C      THE MATRIX IS IN SSP VECTOR STORAGE MODE.
0551 C
0552 C      REMARKS
0553 C      THE HESSENBERG FORM REPLACES THE ORIGINAL MATRIX IN TH
0554 C      ARRAY A.
0555 C
0556 C      SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
0557 C      NONE
0558 C
0559 C      METHOD
0560 C      SIMILARITY TRANSFORMATIONS USING ELEMENTARY ELIMINATIO
0561 C      MATRICES, WITH PARTIAL PIVOTING.
0562 C
0563 C      REFERENCES
0564 C      J.H. WILKINSON - THE ALGEBRAIC EIGENVALUE PROBLEM -
0565 C      CLARENDON PRESS, OXFORD, 1965.
0566 C
0567 C      .....
0568 C
0569 C      SUBROUTINE HSBG(N,A,IA)
0570 C      DIMENSION A(1)
0571 C      DOUBLE PRECISION S
0572 C      L=N
0573 C      NIA=L*IA
0574 C      LIA=NIA-IA
0575 C
0576 C      L IS THE ROW INDEX OF THE ELIMINATION
0577 C
0578 20 IF(L-3) 360,40,40
0579 40 LIA=LIA-IA
0580      L1=L-1
0581      L2=L1-1
0582 C
0583 C      SEARCH FOR THE PIVOTAL ELEMENT IN THE LTH ROW
0584 C

```

```

0585      ISUB=LIA+L
0586      IPIV=ISUB-IA
0587      PIV=ABS(A(IPIV))
0588      IF(L-3) 90,90,50
0589      50 M=IPIV-IA
0590      DO 80 I=L,M,IA
0591      T=ABS(A(I))
0592      IF(T-PIV) 80,80,60
0593      60 IPIV=I
0594      PIV=T
0595      80 CONTINUE
0596      90 IF(PIV) 100,320,100
0597      100 IF(PIV-ABS(A(ISUB))) 180,180,120
0598 C
0599 C      INTERCHANGE THE COLUMNS
0600 C
0601      120 M=IPIV-L
0602      DO 140 I=1,L
0603      J=M+I
0604      T=A(J)
0605      K=LLA+I
0606      A(J)=A(K)
0607      140 A(K)=T
0608 C
0609 C      INTERCHANGE THE ROWS
0610 C
0611      M=L2-M/IA
0612      DO 160 I=L1,NIA,IA
0613      T=A(I)
0614      J=I-M
0615      A(I)=A(J)
0616      160 A(J)=T
0617 C
0618 C      TERMS OF THE ELEMENTARY TRANSFORMATION
0619 C
0620      180 DO 200 I=L,LLA,IA
0621      200 A(I)=A(I)/A(ISUB)
0622 C
0623 C      RIGHT TRANSFORMATION
0624 C
0625      J=-IA
0626      DO 240 I=1,L2
0627      J=J+IA
0628      LJ=L+J
0629      DO 220 K=1,L1
0630      KJ=K+J
0631      KL=K+LLA
0632      220 A(KJ)=A(KJ)-A(LJ)*A(KL)
0633      240 CONTINUE
0634 C
0635 C      LEFT TRANSFORMATION
0636 C

```

```

0637      K=-LA
0638      DO 300 I=1,M
0639      K=K+LA
0640      LK=K+L1
0641      S=A(LK)
0642      LJ=L-IA
0643      DO 280 J=1,L2
0644      JK=K+J
0645      LJ=LJ+IA
0646      280 S=S+A(LJ)*A(JK)*1.0D0
0647      300 A(LK)=S
0648 C
0649 C      SET THE LOWER PART OF THE MATRIX TO ZERO
0650 C
0651      DO 310 I=L,LIA,IA
0652      310 A(I)=0.0
0653      320 L=L1
0654      GO TO 20
0655      360 RETURN
0656      END
0657      SUBROUTINE MLTNX(A,S,M,MDIM)
0658 C
0659 C      SUBROUTINE OBTAINS THE MATRIX MULTIPLICATION OF A AND S AND
0660 C      THE RESULTS IN S.
0661 C
0662      DIMENSION S(MDIM,MDIM),A(MDIM,MDIM)
0663      COMMON/WORK/T(25,25)
0664      DO 10 I=1,M
0665      DO 10 J=1,M
0666      C=0.0
0667      DO 20 K=1,M
0668      20 C=C+A(I,K)*S(K,J)
0669      10 T(I,J)=C
0670      DO 50 I=1,M
0671      DO 50 J=1,M
0672      50 S(I,J)=T(I,J)
0673      RETURN
0674      END
0675      BLOCK DATA WORK
0676      COMMON /WORK/ WO(625)
0677      END
0678  $

```

&FILTR T=00004 IS ON CRO0022 USING 00004 BLKS R=0022

```
0001  FTN4,L
0002      PROGRAM FILTR
0003  C WRITTEN BY E. E. SHERROD
0004  C
0005  C THIS PROGRAM SELECTS THE FILTERING TYPE
0006  C
0007      DIMENSION ILU(5),NAME1(3),NAME2(3),IRTN(5),NAME3(3)
0008      EQUIVALENCE(IRTN(2),RMAX),(IRTN(4),RMIN)
0009      DATA NAME1/2HLF,2HLT,2HR /
0010      DATA NAME2/2HHF,2HLT,2HR /
0011      DATA NAME3/2HSH,2HOW,2H /
0012  C
0013  C GET LU
0014  C
0015      CALL RMPAR(ILU)
0016      IPIXL =0
0017      JPIXL =511
0018      LU=ILU(1)
0019      WRITE(LU,10)
0020  10  FORMAT(" SELECT FILTERING TYPE "/" 1. LINEAR "/" 2. HOMOMORP
0021      READ(LU,*) IFITR
0022      IF(IFITR .EQ. 1) CALL EXEC(23,NAME1,LU,IPIXL,JPIXL,0,0)
0023      CALL RMPAR(IRTN)
0024      IF(IFITR .EQ. 1) GO TO 30
0025      IF(IFITR .EQ. 2) CALL EXEC(23,NAME2,LU,IPIXL,JPIXL,0,0)
0026      CALL RMPAR(IRTN)
0027  30  WRITE(LU,40) RMAX,RMIN
0028  40  FORMAT(" MAX PIXEL = ",F12.2,10X," MIN PIXEL = ",1F12.2)
0029      IX=RMAX-RMIN +0.5
0030      WRITE(LU,50) IX
0031  50  FORMAT(" NUMBER OF GRAY LEVELS = ",I5)
0032      IF(IFITR .EQ. 2)CALL EXEC(23,NAME3,LU,0,511,0,0)
0033      STOP
0034      END
0035      ENDS
```

GNOISE T=00004 IS ON CR00022 USING 00010 BLKS R=0097

```
0001  FTN4,L
0002      PROGRAM NOISE
0003  C
0004      DIMENSION RDATA(512),GNOISE(512),LU(5),IU(5),IBUF(40)
0005  C
0006      INTEGER READL
0007      EQUIVALENCE (RDATA,LU(2)),(LU(2),ILINE),(LU(3),IPIXL),
0008      1 (RDATA(2),RMAX),(RDATA(3),RMIN)
0009      DATA RDATA/512*0.0/
0010  C
0011  C GET INPUT PARAMETERS
0012  C
0013      CALL RMPAR(LU)
0014  C
0015  C SCHEDULE BUILD WORK FILE PROGRAM
0016      CALL EXEC(23,6HBLDF ,IU)
0017  C
0018  C READ WORK FILE HEADER
0019  C
0020      IERR = READL(-1,0,511,RDATA)
0021      IF (IERR .LT. 0) GO TO 999
0022      NLINE=ILINE
0023      NPIXL=IPIXL
0024      PMAX=RMAX
0025      PMIN=RMIN
0026  C
0027  C GET NOISE INFO
0028      WRITE(LU,13)
0029      13 FORMAT(" ENTER NOISE MEAN VALUE _")
0030      READ(LU,*) AM
0031      WRITE(LU,14)
0032      14 FORMAT(" ENTER STANDARD DEVIATION VALUE _")
0033      READ(LU,*) S
0034      IF(S .LE. 0) GO TO 1000
0035  C
0036      DO 100 I=0,NLINE-1
0037      IF (READL(I,0,NPIXL-1,RDATA) .LT. 0) GO TO 999
0038  C
0039  C GET NOISE
0040  C
0041      DO 101 JA=0,51
0042      CALL EXEC(1,8,IBUF,40)
0043      JJ=10*JA
0044      CALL CODE (80)
0045      READ( IBUF,12) (GNOISE(K+JJ),K=1,10 )
0046      12 FORMAT(10F8.5)
0047      101 CONTINUE
```

```
0048 C
0049      DO 90 J =1,NPIXL
0050      RDATA(J)= RDATA(J) +GNOISE(J)*S+AM
0051 600  FORMAT( F20.3)
0052 90   CONTINUE
0053 C
0054 C WRITE SIGNAL + NOISE TO WORK FILE
0055 C
0056      IF(RITEL(I,0,NPIXL-1,RDATA) .LT. 0) GO TO 999
0057      IF(MOD(I,64) .EQ. 0) WRITE(LU,4)
0058 4   FORMAT(" **** ADDING NOISE ****")
0059 100  CONTINUE
0060 C
0061 1000 CALL CLSWF(NLINE,NPIXL,PMAX,PMIN)
0062      CALL CLOSE(IDCBL)
0063      CALL EXEC(6)
0064 999  WRITE(LU,2) IERR
0065 2   FORMAT("FILE ERROR",I7)
0066      END
0067 $
0068 $
```

APPENDIX E

Stability Analysis of Two-Dimensional Digital Recursive Filters

WINSER E. ALEXANDER, MEMBER, IEEE, AND STEVEN A. PRUESS

Abstract—A new approach to the stability problem for the two-dimensional digital recursive filter is presented. The bivariate difference equation representation of the two-dimensional recursive digital filter is converted to a multinput-multoutput (MIMO) system similar to the state-space representation of the one-dimensional digital recursive filter. In this paper, a pseudo-state representation is used and three coefficient matrices are obtained. A general theorem for stability of two-dimensional digital recursive filters is derived and a very useful theorem is presented which expresses sufficient requirements for instability in terms of the spectral radii of these matrices.

I. INTRODUCTION

A two-dimensional digital recursive filter can be characterized by the bivariate difference equation

$$g(m, n) = \sum_{J=0}^L \sum_{K=0}^L a_{JK} f(m-J, n-K) - \sum_{J=0}^L \sum_{\substack{K=0 \\ J+K>0}}^L b_{JK} g(m-J, n-K) \quad (1)$$

Manuscript received October 10, 1975; revised March 7, 1979. This work was supported in part by the U.S. Department of Energy and also by the Naval Air Systems Command.

W. E. Alexander is with the Department of Electrical Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC 27411.

S. A. Pruess is with the Department of Mathematics, The University of New Mexico, Albuquerque, NM 87106.

where the coefficients a_{JK} and b_{JK} are constants [1] and some of these constants may be zero. In general, this form does not require that the corresponding numerator and denominator polynomials for the two-dimensional Z transform of the transfer function both be of degree L . Zeros may be added to form the structure as given in (1). There are two major problems to consider in the design of recursive filters for two-dimensional signal processing: synthesis and stability. The synthesis problem consists of determining the filter coefficients so that the required frequency response is realized. If the resulting filter is to be useful, it must be bounded-input-bounded-output (BIBO) stable. In this paper the stability problem is considered and a new approach to stability analysis for the two-dimensional digital recursive filter is presented.

For the one-dimensional case, there are essentially two methods of determining necessary and sufficient conditions for stability of digital filters: examining regions of analyticity for the characteristic polynomial and by direct evaluation of the characteristics of the impulse response [2]–[4]. In particular, if the system corresponding to the digital filter is represented by a state-space equation, then one can determine stability from the coefficient matrices in the state-space equation [4]. For the two-dimensional case, generalizations of the first method involves examining regions of analyticity for bivariate polynomials [5].

This paper attempts to generalize the second method for the two-dimensional case, i.e., to establish stability by computing the spectral radii of coefficient matrices with real coefficients. The spectral radius of a matrix is the magnitude of the largest magnitude eigenvalue of that matrix.

II. PSEUDO-STATE-SPACE REPRESENTATION

A pseudo-state-space representation of (1) is used in the development of the stability analysis theorems in this paper. This representation is very similar to a state-space representation of the two-dimensional digital recursive filter as defined by Fornasini and Marchesini [6]. The two can be made to be equivalent by letting one of the coefficient matrices in the Fornasini and Marchesini model be the null matrix. The pseudo-state-space representation of the two-dimensional recursive filter is given by

$$\begin{aligned} G_{m,n} &= \bar{B}_1 G_{m,n-1} + \bar{B}_2 G_{m-1,n} + \bar{A} F_{m,n} \\ g(m,n) &= D G_{m,n} \end{aligned} \quad (2)$$

$G_{m,n}$ is a column vector such that its elements are the outputs, $g(m-J, n-K)$ where $0 < J < L$ and $0 < K < L$. Note that $G_{m,n}$ contains all of the outputs that are represented in (1) including $g(m,n)$. Similarly, $F_{m,n}$ is a column vector such that its elements are the inputs, $f(m-J, n-K)$ where $0 < J < L$ and $0 < K < L$.

We can then define matrices \bar{B}_1 , \bar{B}_2 , and \bar{A} [7] such that (1) and (2) are equivalent. The matrices \bar{B}_1 , \bar{B}_2 , and \bar{A} are all of order $(L+1)^2$ by $(L+1)^2$. The vector D is a row vector with $L+1$ elements.

The ordering of the outputs in $G_{m,n}$ and of the inputs in $F_{m,n}$ is not unique. However, the ordering does affect the relative position of the elements of the corresponding coefficient matrices. Also note that $G_{m-1,n}$ and $G_{m,n-1}$ have elements in common. Where this occurs, the corresponding elements of \bar{B}_1 and \bar{B}_2 can be divided such that the magnitude of each is no larger than that of the corresponding b_{JK} or one as appropriate. It is convenient to consistently divide equally and choose a particular ordering scheme for $G_{m,n}$.

Example

Consider the two-dimensional digital recursive filter with bivariate difference equation given by

$$\begin{aligned} g(m,n) &= a_{00}f(m,n) + a_{10}f(m-1,n) + a_{01}f(m,n-1) \\ &\quad + a_{11}f(m-1,n-1) - b_{10}g(m-1,n) \\ &\quad - b_{01}g(m,n-1) - b_{11}g(m-1,n-1). \end{aligned} \quad (3)$$

For this example, with $G_{m,n}$ and $F_{m,n}$ given in transpose form, we have

$$G_{m,n} = [g(m,n) \ g(m-1,n) \ g(m,n-1) \ g(m-1,n-1)]^T \quad (4)$$

$$F_{m,n} = [f(m,n) \ f(m-1,n) \ f(m,n-1) \ f(m-1,n-1)]^T \quad (5)$$

$$\begin{aligned} \bar{B}_1 &= \begin{bmatrix} -b_{10} & 0 & -\frac{1}{2}b_{11} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \end{bmatrix} \\ \bar{B}_2 &= \begin{bmatrix} -b_{01} & -\frac{1}{2}b_{11} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \end{bmatrix} \quad (6) \\ \bar{A} &= \begin{bmatrix} a_{00} & a_{10} & a_{01} & a_{11} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (7) \end{aligned}$$

III. STABILITY ANALYSIS

The stability analysis herein will be confined to the linear shift invariant (LSI) two-dimensional discrete system. Such a system is BIBO stable if and only if the discrete impulse response of the system, $h(m,n)$, is absolutely summable, i.e., $\sum_{m,n=-\infty}^{\infty} |h(m,n)| < \infty$ [1].

Let us define the particular vector H_{JK} as that input vector which represents a single unit sample at the (J,K) position of the two-dimensional data array with all other inputs samples zero. Let us further define the initial condition vectors, $G_{J-1,K}$ and $G_{J,K-1}$, as null vectors. Then for $m=J$ and $n=K$, (2) reduces to

$$\begin{aligned} G_{J,K} &= \bar{A} H_{J,K} \\ h(J,K) &= D G_{J,K}. \end{aligned} \quad (8)$$

Define the term $C(\bar{B}_1^J, \bar{B}_2^K)$ as the sum of all product terms involving all permutations of \bar{B}_1 as a factor J times and \bar{B}_2 as a factor K times. It is helpful to note that if \bar{B}_1 and \bar{B}_2 commute, then

$$C(\bar{B}_1^J, \bar{B}_2^K) = \binom{J+K}{K} \bar{B}_1^J \bar{B}_2^K = (J+K)! \bar{B}_1^J \bar{B}_2^K / (J!K!).$$

In general, the matrices do not commute. Therefore, we give as an example $C(\bar{B}_1^2, \bar{B}_2^1) = \bar{B}_1^2 \bar{B}_2 + \bar{B}_1 \bar{B}_2 \bar{B}_1 + \bar{B}_2 \bar{B}_1^2$.

Lemma 1

The contribution to the output vector, $G_{m,n}$, for a single input vector, $H_{J,K}$, which corresponds to a unit impulse at the (J,K) position where $J < m$ and $K < n$, is given by $G_{m,n} = C(\bar{B}_1^{m-J}, \bar{B}_2^{n-K}) \bar{A} H_{J,K}$ for the LSI system represented by (2).

The proof of Lemma 1 is given in the Appendix. Lemma 1 provides a convenient means of finding the output of the two-dimensional digital recursive filter for all values of m and n when the filter is excited by a single input at any point in the array. Since the filter is linear and shift invariant, we can use the principle of superposition to find the output for any particular sequence of inputs. Thus the unit impulse response of the filter is given

by

$$G_{m,n} = C(\bar{B}_1^m, \bar{B}_2^n) \bar{A} H_{0,0} \quad (9)$$

$$h(m,n) = DG_{m,n} = DC(\bar{B}_1^m, \bar{B}_2^n) \bar{A} H_{0,0} \quad (10)$$

Lemma 2

Given the discrete LSI system represented by (2) for which the corresponding transfer function has mutually prime numerator and denominator polynomials. If the contribution to the output vector $G_{m,n}$ by a bounded sequence of input vectors $F_{J,K}$ where $0 \leq J \leq m$ and $0 \leq K \leq n$ can be expressed by $G_{m,n} = \bar{Q}^m \bar{A} F_{J,K}$ or $G_{m,n} = \bar{Q}^n \bar{A} F_{J,K}$, then the system is unstable if $\rho(\bar{Q})$, the spectral radius of \bar{Q} , is greater than one. The proof of Lemma 2 is given in the Appendix.

Theorem 1

The discrete LSI system represented by (2) is stable if and only if for at least one matrix norm

$$S = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \|DC(\bar{B}_1^m, \bar{B}_2^n) \bar{A} H_{0,0}\| < \infty. \quad (11)$$

Theorem 1 follows directly from (10) and the requirement that the discrete impulse response be absolutely summable. Since $h(m,n)$ is a scalar, its matrix norm is equivalent to its absolute value and the proof of Theorem 1 is obvious.

Theorem 2

The discrete LSI system represented by (2) and for which the numerator and denominator polynomials of the corresponding transfer function are mutually prime is unstable if any one of the spectral radii $\rho(\bar{B}_1)$, $\rho(\bar{B}_2)$, or $\rho(\bar{B}_1 + \bar{B}_2)$ is greater than or equal to one. The proof of Theorem 2 is given in the Appendix.

In the practical application of two-dimensional digital recursive filters, any filter with $\rho(\bar{B}_1)$, $\rho(\bar{B}_2)$, or $\rho(\bar{B}_1 + \bar{B}_2)$ equal to one can be considered to be unstable and should be avoided [8]. Goodman [5] has shown by clever examples that two-dimensional filters with nonessential singularities of the second kind on the unit bidisc may be stable. Such a filter may have $\rho(\bar{B}_1)$, $\rho(\bar{B}_2)$, or $\rho(\bar{B}_1 + \bar{B}_2)$ equal to one. However, roundoff errors and coefficient truncation would prevent satisfactory performance by such a filter for most applications.

IV. CONCLUSIONS

In this paper, a new approach to stability analysis of two-dimensional digital recursive filters has been presented. Theorems have been presented which can be used in the practical application of this approach. The authors feel that it is important to note that no known unstable filter has been found in this research effort which did not have either $\rho(\bar{B}_1)$, $\rho(\bar{B}_2)$, or $\rho(\bar{B}_1 + \bar{B}_2)$ greater than or equal to one. One is lead to conjecture that for a large class of filters, any filter in the class is stable if the subject spectral radii are all less than one. However, the proof of this is not trivial.

Several other theorems relating to sufficient conditions for stability have been found [7]. However, it has been shown that these constraints are too restrictive for general use. That is, useful stable filters can be found which do not satisfy the corresponding sufficient conditions for stability.

Computer algorithms are readily available to find the spectral radius of a matrix with real coefficients. Thus Theorem 2 presents a convenient and easily implemented technique to assess the stability of two-dimensional digital recursive filters.

APPENDIX

In this Appendix, the proofs for Lemmas 1 and 2 and Theorem 2 are given. When a specific norm is not given, any convenient norm is appropriate.

A1. Proof of Lemma 1

We proceed with a proof by induction. If we use (2) and (8) to obtain $G_{J+1,K}$, $G_{J,K+1}$, and $G_{J+1,K+1}$ for input vector $H_{J,K}$ and if all initial condition vectors are null vectors, we obtain

$$\left. \begin{aligned} G_{J+1,K} &= \bar{B}_1 G_{J,K} = \bar{B}_1 \bar{A} H_{J,K} \\ G_{J,K+1} &= \bar{B}_2 G_{J,K} = \bar{B}_2 \bar{A} H_{J,K} \\ G_{J+1,K+1} &= \bar{B}_1 G_{J,K+1} + \bar{B}_2 G_{J+1,K} = (\bar{B}_1 \bar{B}_2 + \bar{B}_2 \bar{B}_1) \bar{A} H_{J,K} \end{aligned} \right\} \quad (A1)$$

If we use Lemma 1, we obtain

$$\left. \begin{aligned} G_{J+1,K} &= C(\bar{B}_1^1, \bar{B}_2^0) \bar{A} H_{J,K} = \bar{B}_1 \bar{A} H_{J,K} \\ G_{J,K+1} &= C(\bar{B}_1^0, \bar{B}_2^1) \bar{A} H_{J,K} = \bar{B}_2 \bar{A} H_{J,K} \\ G_{J+1,K+1} &= C(\bar{B}_1^1, \bar{B}_2^1) \bar{A} H_{J,K} = (\bar{B}_1 \bar{B}_2 + \bar{B}_2 \bar{B}_1) \bar{A} H_{J,K} \end{aligned} \right\} \quad (A2)$$

Thus for any arbitrary m and n such that $m > J$ and $n > K$, we can use (2) to write

$$G_{m+1,n} = \bar{B}_1 G_{m,n} + \bar{B}_2 G_{m+1,n-1}. \quad (A3)$$

Then using (9) to find expressions for $G_{m,n}$ and $G_{m+1,n-1}$, we have

$$\begin{aligned} G_{m+1,n} &= [\bar{B}_1 C(\bar{B}_1^{m-J}, \bar{B}_2^{n-K}) \\ &\quad + \bar{B}_2 C(\bar{B}_1^{m-J+1}, \bar{B}_2^{n-K-1})] \bar{A} H_{J,K}. \end{aligned} \quad (A4)$$

Consider the term, $C(\bar{B}_1^J, \bar{B}_2^K)$. All of the products in the term either have \bar{B}_1 as the first factor or \bar{B}_2 as the first factor. If \bar{B}_1 is the first factor, we must postmultiply by the sum of all possible products such that the power of \bar{B}_1 is decreased by one. If \bar{B}_2 occurs as the first factor, we must post-multiply by the sum all possible products such that the power of \bar{B}_2 is decreased by one. We conclude that

$$C(\bar{B}_1^J, \bar{B}_2^K) = \bar{B}_1 C(\bar{B}_1^{J-1}, \bar{B}_2^K) + \bar{B}_2 C(\bar{B}_1^J, \bar{B}_2^{K-1}) \quad (A5)$$

for all J and K , such that both J and K are greater than or

equal to one. It follows directly that

$$G_{m+1,n} = C(\bar{B}_1^{m+1-J}, \bar{B}_2^{n-K}) \bar{A} H_{J,K}. \quad (\text{A6})$$

Similarly from (2) we can write

$$G_{m,n+1} = \bar{B}_1 G_{m-1,n+1} + \bar{B}_2 G_{m,n}. \quad (\text{A7})$$

Using (9) to find expressions for $G_{m-1,n+1}$ and $G_{m,n}$, we have

$$G_{m,n+1} = [\bar{B}_1 C(\bar{B}_1^{m-1-J}, \bar{B}_2^{n+1-K}) + \bar{B}_2 C(\bar{B}_1^{m-J}, \bar{B}_2^{n-K})] \bar{A} H_{J,K}. \quad (\text{A8})$$

It follows that

$$G_{m,n+1} = C(\bar{B}_1^{m-J}, \bar{B}_2^{n+1-K}) \bar{A} H_{J,K}. \quad (\text{A9})$$

Finally, from (2) we obtain

$$G_{m+1,n+1} = \bar{B}_1 G_{m,n+1} + \bar{B}_2 G_{m+1,n}. \quad (\text{A10})$$

Using Lemma 1 to express $G_{m,n+1}$ and $G_{m+1,n}$, we obtain

$$G_{m+1,n+1} = [\bar{B}_1 C(\bar{B}_1^{m-J}, \bar{B}_2^{n+1-K}) + \bar{B}_2 C(\bar{B}_1^{m+1-J}, \bar{B}_2^{n-K})] \bar{A} H_{J,K}. \quad (\text{A11})$$

It follows from (A5) and (A11) that

$$G_{m+1,n+1} = C(\bar{B}_1^{m+1-J}, \bar{B}_2^{n+1-K}) \bar{A} H_{J,K} \quad (\text{A12})$$

and Lemma 1 holds.

A2. Proof of Lemma 2

In the proof of Lemma 2, we shall show that if the response to a particular sequence of input vectors can be represented as given in Lemma 2, then the system is unstable if $\rho(\bar{Q}) > 1$ [9].

Define the eigenvalue corresponding to the spectral radius of \bar{Q} as λ_Q and the corresponding eigenvector as P_Q . Then if the system transfer function has mutually prime numerator and denominator polynomials we can select a sequence of input vectors such that

$$\bar{A} F_{J,n} = \epsilon P_Q + R_{J,n} \quad \text{for all } J \text{ and } n. \quad (\text{A13})$$

where ϵ is an arbitrary nonzero finite constant and $R_{J,n}$ is not in the direction of P_Q . We then have

$$G_{m,n} = \bar{Q}^m \bar{A} F_{J,n} = \epsilon \bar{Q}^m P_Q + \bar{Q}^m R_{J,n}. \quad (\text{A14})$$

Then since λ_Q is the eigenvalue corresponding to the spectral radius, the norm of $G_{m,n}$ is dominated by the term $\epsilon \bar{Q}^m P_Q$ in the limit as m approaches infinity. Thus

$$S = \lim_{m \rightarrow \infty} \|G_{m,n}\| = \lim_{m \rightarrow \infty} \|\epsilon \bar{Q}^m P_Q\| = \lim_{m \rightarrow \infty} \|\epsilon \lambda_Q^m P_Q\|. \quad (\text{A15})$$

Note that S is infinite if λ_Q is greater than one and Lemma 2 holds.

A3. Proof of Theorem 2

For this proof, we show that we can find a particular sequence of inputs that give unbounded output if any one of the spectral radii specified in Theorem 2 is greater than one.

From Lemma 1 the output from a single arbitrary bounded input at the (J, K) position can be given by

$$G_{M,N} = f(J, K) C(\bar{B}_1^{M-J}, \bar{B}_2^{N-K}) \bar{A} H_{J,K} \\ g(M, N) = D G_{M,N} \quad (\text{A16})$$

where $f(J, K)$ is the scalar input at the (J, K) position. If we let $K = N$ and $J = 0$ in (A16), we have

$$G_{M,N} = f(0, N) C(\bar{B}_1^M, \bar{B}_2^0) \bar{A} H_{0,N} = f(0, N) \bar{B}_1^M \bar{A} H_{0,N}. \quad (\text{A17})$$

If we apply Lemma 2, we see that the system is unstable if $\rho(\bar{B}_1) > 1$. If we let $J = M$ and $K = 0$ in (A16), we have

$$G_{M,N} = f(M, 0) C(\bar{B}_1^0, \bar{B}_2^N) \bar{A} H_{M,0} = f(M, 0) \bar{B}_2^N \bar{A} H_{M,0}. \quad (\text{A18})$$

If we apply Lemma 2, we see that the system is unstable if $\rho(\bar{B}_2) > 1$.

If we use a particular sequence of inputs $f(J, M - J)$ for $0 \leq J \leq M$ where all $f(J, M - J)$ are bounded and equal. Using the principle of superposition and (A16) we have

$$G_{M,M} = \sum_{J=0}^M f(J, M-J) C(\bar{B}_1^{M-J}, \bar{B}_2^J) \bar{A} H_{J,M-J}. \quad (\text{A19})$$

Since all inputs are equal, we can write

$$G_{M,M} = f(0, M) \left[\sum_{J=0}^M C(\bar{B}_1^{M-J}, \bar{B}_2^J) \right] \bar{A} H_{0,M} \quad (\text{A20})$$

$$G_{M,M} = f(0, M) (\bar{B}_1 + \bar{B}_2)^M \bar{A} H_{0,M} \quad (\text{A21})$$

since

$$(\bar{B}_1 + \bar{B}_2)^M = \sum_{J=0}^M C(\bar{B}_1^{M-J}, \bar{B}_2^J) \quad (\text{A22})$$

whether or not \bar{B}_1 and \bar{B}_2 commute. If we apply Lemma 2, we see that the system is unstable if $\rho(\bar{B}_1 + \bar{B}_2) > 1$ and Theorem 2 holds.

ACKNOWLEDGMENT

The author wishes to thank E. I. Jury of the Department of Electrical Engineering, University of California, Berkeley, and the reviewers of this paper for supplying helpful comments, suggestions and references.

REFERENCES

- [1] Lawrence R. Rabiner and Bernard Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975, pp. 442-455.
- [2] Samuel Stearns, *Digital Signal Analysis*. Hayden Publishing Company, 1975, p. 134.
- [3] E. I. Jury, "Theory and application of inners," *Proc. IEEE*, vol. 63, pp. 1044-1068, 1975.
- [4] Katsuhiko Ogata, *State Space Analysis of Control Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1967, p. 487.
- [5] Dennis Goodman, "Some stability properties of two-dimensional linear shift-invariant digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 201-208, 1977.
- [6] E. Fornasini and G. Marchesini, "State space realization theory for two-dimensional filters," *IEEE Trans. Automat. Contr.*, vol. AC-21, pp. 484-492, Aug. 1976.
- [7] Winzer E. Alexander, "Stability and synthesis of two-dimensional digital recursive filters," Ph.D. dissertation, Univ. of New Mexico, Albuquerque, NM, 1974 (University Microfilms, Ann Arbor, MI).

- [8] N. K. Bose, "Problems and progress in multidimensional system theory", *Proc IEEE*, vol. 65, pp. 724-840, 1977.
- [9] Alston S. Householder, *The Theory of Matrices in Numerical Analysis*, New York, NY: Blaupunkt, 1964, ch. 2.



Wimber E. Alexander (M'70) was born in Columbia, NC, on August 19, 1942. He received the B.S. degree in electrical engineering from North Carolina Agricultural and Technical State University, in 1964. He received the M.S. degree in engineering and the Ph.D. degree in electrical engineering from the University of New Mexico, Albuquerque, in 1966 and 1974, respectively. He is currently Professor and Chairman of the Department of Electrical Engineering at North Carolina A and T State University. Before joining A and T he was a member of the technical staff at Sandia Laboratories from January 1970 to January 1976 and from June 1964 to July

1966. He served as an officer in the U.S. Air Force from July 1966 to December 1969 and was honorably discharged as a Captain in December 1969. His research interests are digital signal processing and image processing.



Steven A. Pruess was born in Cedar Rapids, IA, on June 8, 1944. He received the B.S. degree in mathematics from Iowa State University, Ames, in 1966, and the M.S. and Ph.D. degrees in computer science from Purdue University, Lafayette, IN, in 1968 and 1970, respectively.

Since 1970 he has been a member of the faculty of the Department of Mathematics and Statistics at the University of New Mexico, Albuquerque, where he is currently Associate Professor. His research interests are in numerical analysis and approximation theory.

Dr. Pruess is a member of the Society for Industrial and Applied Mathematics, and the American Mathematical Society.

STABILITY ANALYSIS OF TWO DIMENSIONAL DIGITAL RECURSIVE FILTERS

Winston E. Alexander

Department of Electrical Engineering
North Carolina A&T State University

ABSTRACT

This paper presents a new procedure for stability analysis of two dimensional recursive digital filters. A matrix recursive equation which is similar to the state space representation of the one dimensional digital recursive filter is formulated. This matrix recursive equation is used to assess stability of the two dimensional digital recursive filter in terms of the spectral radii of the coefficient matrices.

Examples of the use of this technique to assess stability of two dimensional digital recursive filters are given. It is demonstrated that this technique reduces the stability analysis problem to examining the spectral radii of matrices with constant coefficients.

INTRODUCTION

A causal two dimensional digital recursive filter may be represented by the bivariate difference equation

$$g(m,n) = \sum_{J=0}^L \sum_{K=0}^L a_{JK} f(m-J, n-K) - \sum_{J=0}^L \sum_{K=0}^L b_{JK} g(m-J, n-K) \quad (1)$$

J+K>0

where some of the coefficients a_{JK} and b_{JK} may be zero. Such a filter uses feedback of past output values to calculate the current output. Therefore, it may be bounded input-bounded output (BIBO) unstable. That is, the output may not be bounded for a given bounded input. This paper considers this stability problem and present a simple technique to assess stability of two dimensional recursive digital filters.

The Matrix Recursive Form

The bivariate difference equation represented by (1) can be described by the matrix recursive equation

$$G_{m,n} = B_1 G_{m-1,n} + B_2 G_{m,n-1} + A F_{m,n} \quad (2)$$

where $G_{m,n}$ is a column vector made up of all outputs in (1), $F_{m,n}$ is a column vector made up of all inputs in (1) and the matrices B_1 , B_2 and A are appropriate matrices to make (1) and (2) equivalent. The matrices B_1 , B_2 and A are all of order $(L+1)^2$ by $(L+1)^2$. The current output is then given by $g(m,n) = DG_{m,n}$ where D is a row vector with $(L+1)$ elements.

The ordering of the outputs in $G_{m,n}$ and of the inputs in $F_{m,n}$ is not unique. However, the ordering does affect the relative position of the elements of the corresponding B_1 and B_2 matrices. Also note that there are identical elements in B_1 and B_2 . Where this occurs, the corresponding elements of B_1 and B_2 can be divided such that the magnitude of each is no longer than that of the corresponding b_{JK} or one as appropriate. It is convenient to consistently divide each and choose a particular ordering scheme.

Example 1

Consider the recursive digital filter with bivariate difference equation given by

$$g(m,n) = f(m,n) - b_{10}g(m-1,n) - b_{01}g(m,n-1) - b_{11}g(m-1,n-1) \quad (3)$$

For this example, we have

$$G_{m,n} = \begin{bmatrix} g(m,n) \\ g(m-1,n) \\ g(m,n-1) \\ g(m-1,n-1) \end{bmatrix}; F_{m,n} = \begin{bmatrix} f(m,n) \\ f(m-1,n) \\ f(m,n-1) \\ f(m-1,n-1) \end{bmatrix}$$

$$\underline{B}_1 = \begin{bmatrix} -b_{10} & 0 & b_{11} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} : \underline{B}_2 = \begin{bmatrix} -b_{01} & -b_{11} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (5)$$

and

$$\bar{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (6)$$

Stability Analysis

For the one dimensional case, there are essentially two methods of determining necessary and sufficient conditions for stability; examining regions of analyticity for the characteristic polynomial and by direct evaluation of the characteristics of the impulse response [1,2,3]. In particular, if the filter is represented as a state space equation, then one can determine stability from the coefficient matrices in the state space equation [4]. The usual approach for stability analysis of two dimensional digital recursive filters involves examining regions of analyticity for bivariate polynomials [4] which is computational feasible only for very simple filters. This paper represents an attempt to generalize the second method for the two dimensional case, i.e. to establish stability by computing the spectral radii of coefficient matrices with real coefficients.

The following theorems relating to stability analysis of two dimensional digital recursive filters have been developed [5]. Space will not allow proof of the theorems in this paper. The reader is referred to reference [5] for further details.

Theorem 1

The linear space invariant (LSI) two dimensional digital recursive filter represented by (2) and for which the numerator and denominator polynomials of the corresponding transfer function are mutually prime is unstable if any one of the spectral radii $\rho(\underline{B}_1)$, $\rho(\underline{B}_2)$, $\rho(\underline{B}_1 + \underline{B}_2)$ is greater than or equal to one. The spectral radius of a given matrix is the magnitude of the largest magnitude eigenvalue of that matrix).

Theorem 2

The LSI two dimensional digital recursive filter represented by (2) is stable if the spectral radius of the matrix made up of the sum of the magnitude of the coefficients of \underline{B}_1 and \underline{B}_2 is less than one ($\rho[\text{abs}(\underline{B}_1) + \text{abs}(\underline{B}_2)] < 1$).

Theorem 3

There is a particular permutation matrix \underline{S} [5] such that if $\rho(\underline{B}_1)$, $\rho(\underline{B}_2)$, $\rho(\underline{B}_1 + \underline{B}_2)$ are all less than one, then the LSI digital recursive filter is stable if both $\rho(\underline{B}_1 \underline{S})$ and $\rho(\underline{B}_2 \underline{S})$ are less than one-half.

Conjecture

If the coefficients of (1) are symmetric such that $b_{JK} = b_{KJ}$ for all J and K , then the LSI recursive digital filter described by (2) and for which the numerator and denominator polynomials of the corresponding transfer function are mutually prime is stable if and only if $\rho(\underline{B}_1)$, $\rho(\underline{B}_2)$, and $\rho(\underline{B}_1 + \underline{B}_2)$ are all less than one.

Example 2

From Theorem 1, we obtain the results that the filter represented by (3) is unstable if $|\underline{b}_{01}| \geq 1$, $|\underline{b}_{10}| \geq 1$, or if $\max \left[\left| \frac{-(\underline{b}_{10} + \underline{b}_{01})}{2} \pm \frac{1}{2} \sqrt{(\underline{b}_{10} + \underline{b}_{01})^2 - 4\underline{b}_{11}} \right| \right] \geq 1 \quad (7)$

Example 3

Consider the example (also used by Shanks [6]) where the bivariate difference equation is given by $g(m,n) = f(m,n) + 0.95 g(m-1,n) + 0.95 g(m,n-1) - 0.5 g(m-1,n-1) \quad (8)$

If we apply Theorem 1, we obtain $\rho(\underline{B}_1) = 0.95$, $\rho(\underline{B}_2) = 0.95$ and $\rho(\underline{B}_1 + \underline{B}_2) = 1.584$. Thus it follows that this filter is unstable.

Example 4

Consider the example used by Read and Treitel [7] with bivariate difference equation given by

$$\begin{aligned} g(m,n) &= f(m,n) + 0.75 g(m-1,n) - 1.5 g(m,n-1) \\ &- 0.9 g(m-2,n) - 1.2 g(m,n-2) - 1.3 g(m-2,n-1) \\ &- 0.9 g(m-1,n-2) - 0.5 g(m-2,n-2) \end{aligned} \quad (9)$$

If we apply Theorem 1, we obtain $\rho(\underline{B}_1) = 1.095$, $\rho(\underline{B}_2) = 0.949$ and $\rho(\underline{B}_1 + \underline{B}_2) = 1.284$. We conclude as did Read and Treitel that this filter is unstable.

Example 5

Consider the example by Huang [8] with difference equation given by

$$\begin{aligned} g(m,n) &= f(m,n) - 0.5 g(m-1,n) - 0.5 g(m,n-1) \\ &- 0.25 g(m-1,n-1) - 0.25 g(m-2,n) - 0.25 g(m,n-2) \end{aligned} \quad (10)$$

If we apply Theorem 3, we obtain $\rho(\underline{B}_1) = 0.5$, $\rho(\underline{B}_2) = 0.5$, $\rho(\underline{B}_1 + \underline{B}_2) = 0.866$; $\rho(\underline{B}_1 S) = \rho(\underline{B}_2 S) = 0.35355$. Therefore, we conclude that this filter is stable. This filter was verified to be stable by Maria and Fahmy [8].

Example 6

Consider the example used by Huang [8] with difference equation given by

$$g(m,n) = f(m,n) - b_{10} g(m-1,n) - b_{01} g(m,n-1) \quad (11)$$

If we apply Theorem 2, it is interesting to note that we get the same sufficient condition for stability as obtained by Huang:

$$|b_{10}| + |b_{01}| < 1 \quad (12)$$

In considering more complex examples, it is convenient to present the coefficients b_{jk} in matrix form. Let the matrix \underline{V} be made up of the elements V_{jk} for row j and column k where

$V_{jk} = b_{j-1, k-1}$. For example, the \underline{V} matrix corresponding to example (1) is given by

$$\underline{V} = \begin{bmatrix} 1.0 & b_{01} \\ b_{10} & b_{11} \end{bmatrix} \quad (13)$$

Note that \underline{V} is of order $(L+1)$ by $(L+1)$.

Example 7

Consider the example used by Read and Treitel where \underline{V} is given by

$$\underline{V} = \begin{bmatrix} 1.0 & 1.5 & -1.9 & -0.8 & 1.1 \\ 1.4 & 2.1 & -2.6 & -1.1 & 1.5 \\ -1.8 & -2.4 & 3.3 & 1.3 & -1.6 \\ -0.7 & -0.9 & 1.1 & 0.5 & -0.8 \\ -0.9 & 1.3 & -1.6 & -0.6 & 1.0 \end{bmatrix} \quad (14)$$

For this example, $\rho(\underline{B}_1) = 2.169$; $\rho(\underline{B}_2) = 2.104$ and $\rho(\underline{B}_1 + \underline{B}_2) = 2.599$. Thus Read and Treitel's conclusion that this filter is unstable is verified.

CONCLUSION

A new procedure for assessing stability of two dimensional recursive digital filters has been presented. The formulation of the \underline{B}_1 and \underline{B}_2 matrices is very simple and straightforward and the matrices are sparse (mostly zeros). Computer algorithms are readily available to obtain the spectral radius of a matrix with real coefficients. Thus stability analysis is greatly simplified with respect to methods which have previously been presented.

It is also important to note that in this research effort all known unstable filters have been detected as being unstable when Theorem 1 was applied. We surmise that for a very large class of filters, any filter within the class not detect-

ed as being unstable after applying Theorem 1 is stable. Research continues to prove or disprove this conjecture.

REFERENCES

- [1] Samuel Stearns, Digital Signal Analysis, Hayden Publishing Company, 1975, p. 134.
- [2] E. I. Jury, "Theory and Applications of Inverses", Proc. IEEE, Vol. 63, No. 7, 1975, pp 1044-1068.
- [3] Katsuhiko Ogata, State Space Analysis of Control Systems, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1967, p. 487.
- [4] Dennis Goodman, "Some Stability Properties of Two Dimensional Linear-Shift Invariant Digital Filters", IEEE Transactions on Circuits and Systems, Vol CAS-24, No. 4, 1977, pp 201-208.
- [5] Winzer E. Alexander, Stability and Synthesis of Two-Dimensional Digital Recursive Filters, Ph.D. Dissertation, University of New Mexico, Albuquerque, N. M., 1974 (Available from University Microfilms, Ann Arbor, Mich.).
- [6] J. L. Shanks, Sven Treitel and J. H. Justice, "Stability and Synthesis of Two Dimensional Recursive Filters", IEEE Transactions on Audio and Electroacoustics, Vol. Au-20, No. 2, 1972 pp 115-128.
- [7] Randel R. Read and Sven Treitel, "The Stabilization of Two Dimensional Recursive Filters Via the Discrete Hilbert Transform", IEEE Transactions on Geoscience Electronics, Vol. GE-11, No. 3, 1973, pp 153-160.
- [8] G. A. Maria and M. M. Fahmy, "On the Stability of Two Dimensional Digital Filters", IEEE Transactions on Audio and Electroacoustics, Vol. Au-21, 1973, pp 470-472.
- [9] Thomas S. Huang, "Stability of Two Dimensional Recursive Filters", IEEE Transactions on Audio and Electroacoustics, Vol. Au-20, No. 2, 1972, pp 158-163.

Two Dimensional Digital Filters for Subjective Image Processing

WINSER E. ALEXANDER AND EARNEST E. SHERROD

DEPARTMENT OF ELECTRICAL ENGINEERING

NORTH CAROLINA AGRICULTURAL AND TECHNICAL STATE UNIVERSITY

Abstract

This paper presents a design technique for designing approximately circularly symmetric lowpass, highpass, bandpass, high frequency boost and low frequency boost digital filters for subjective image processing applications. An approach is used which parallels the use of the Butterworth, Chebychev or other type of polynomial approximations to obtain one dimensional lowpass digital recursive filters. The other filter designs are then derived from the lowpass filter design. The designed filters are very close to being circularly symmetric for a wide range of critical frequencies. In the design procedure, the squared magnitude characteristic of the desired circularly symmetric filter is chosen in the Laplace Transform domain. The bilinear transformation is then used to map the squared magnitude characteristic into the two dimensional ZW-Transform domain. A pseudo-state space representation for the corresponding two dimensional ZW-Transform is obtained. The eigenvalues with magnitudes less than one are then used as roots of a denominator polynomial with distinct roots to form the ZW-Transform of the stable two dimensional digital filter.

1.0 INTRODUCTION

There are basically two types of image processing: subjective image processing and image correction. Subjective image processing involves the modification of an image in some way to improve the ability of the observer to obtain information or to improve the appearance of the image. Image correction involves the removal of noise or other errors in the image caused by the system producing the image. This paper primarily addresses the design of digital filters for use in subjective image processing.

The user interested in subjective image processing typically desires a variety of filters that can be applied based upon experience or a preliminary evaluation of the subject image. He then wants to observe the results of this filtering operation and make adjustments in the filter parameters before filtering again. Therefore, a computationally efficient algorithm is desirable and fast turn around is vital.

The two dimensional recursive digital filter is a good choice to meet these requirements [1]. The size of the image is not constrained to powers of integers and the number of computations per pixel does not increase as the size of the image is increased. In addition, the image is processed by row which is the normal mode for storage of images on tape or disk.

The common techniques of edge enhancement, contrast enhancement, dynamic range compression, etc. may be accomplished with recursive digital filters. These applications involve lowpass,

highpass, bandpass, high boost and low boost digital filters. This paper presents a design technique which can be used to design approximately circularly symmetric recursive digital filters.

2.0 MATHEMATICAL THEORY

The theoretical basis for the two dimensional ZW-Transform [2] involves the theory for sample data systems. Given discrete samples of a two dimensional function, $f(x,y)$ with sampling increments X and Y respectively, the ZW-Transform for the function is defined by

$$F(z,w) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(mX, nY) z^{-m} w^{-n} \quad (2.1)$$

If the function is an image, then (2.1) reduces to the case where m and n have no negative values and the range of m and n is finite. We further restrict the problem to the case where X and Y are constants. Then, if we use the notation $f(m,n)$ to represent $f(mX, nY)$, we have

$$F(z,w) = \sum_{m=0}^{M} \sum_{n=0}^{N} f(m,n) z^{-m} w^{-n} \quad (2.2)$$

as the ZW-Transform for the image function, $f(m,n)$, which has $(M+1)$ columns and $(N+1)$ rows.

Consider the case where we have an input image with samples $f(m,n)$ and we wish to filter this image to obtain an output image with corresponding samples, $g(m,n)$. The samples of the impulse response of the desired filter are given by $h(m,n)$. The range of m and n for the output is the same as for the input. Thus, the ZW-Transform of $g(m,n)$ is given by

$$G(z,w) = \sum_{m=0}^{M} \sum_{n=0}^{N} g(m,n) z^{-m} w^{-n} \quad (2.3)$$

If we restrict the impulse response such that m and n cannot be negative (a causal system), we can write the ZW-Transform for the impulse response as

$$H(z,w) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} h(m,n) z^{-m} w^{-n} \quad (2.4)$$

In general, the ZW-Transform for the impulse response is an infinite series. In order to implement the spatial domain filter, we must find a closed form expression for $H(z,w)$ such that

$$H(z,w) = \frac{\sum_{j=0}^L \sum_{k=0}^L a(j,k)z^{-m}w^{-n}}{\sum_{j=0}^L \sum_{k=0}^L b(j,k)z^{-m}w^{-n}} \quad (2.5)$$

The convolution property of the ZW-Transform gives the relationship resulting from the convolution of $t(m,n)$ and $h(m,n)$ which is the filtering process

$$C(z,w) = H(z,w)F(z,w) \quad (2.6)$$

If we use the closed form of $H(z,w)$ and restrict $b(0,0)$ to be equal to one and write the resulting equation for a single output value $g(m,n)$, we obtain the difference equation

$$g(m,n) = \sum_{j=0}^L \sum_{k=0}^L a(j,k)f(m-j,n-k) - \sum_{j=0}^L \sum_{k=0, k \neq j}^L b(j,k)g(m-j,n-k) \quad (2.7)$$

If L is relatively small (in practice, L is usually less than 10 for recursive digital filters), equation (2.7) represents a very efficient algorithm for filtering images. Equations (2.5) and (2.7) may also represent a nonrecursive filter if all $b(j,k)$ except $b(0,0)$ are equal to zero.

3.0 STABILITY ANALYSIS

Nonrecursive digital filters are inherently stable. Since there is no feedback of past output values, the impulse response has finite duration. Each output value is a finite sum which is always bounded if the input is bounded.

The stability problem for one dimensional digital recursive filters is straight forward. The roots of the denominator polynomial in the closed form of the one dimensional Z-Transform for the filter impulse response function must have magnitudes less than one. Stability analysis is therefore reduced to finding roots of n th degree polynomials with real, constant coefficients [3]. Stability analysis is not straight forward for the two dimensional problem because a two variable polynomial is not generally factorable into distinct roots. When the polynomial in the denominator of the two dimensional Z-Transform of the impulse response is factorable into distinct roots, the stability analysis procedure is the same as for the one dimensional problem.

The two dimensional stability problem is very complicated if the polynomial in the denominator is not factorable into distinct roots [4]. Efforts by other researchers have been directed toward examining regions of roots for two variable polynomials. An alternate method of assessing stability for one dimensional digital recursive filters is to make a state space representation of the filter [5]. Then the filter is stable if the eigenvalues of the state transition matrix all have magnitudes less than one. Previous research has been directed toward developing the two dimensional equivalent of this procedure [6,7]. A pseudo-state variable representation is chosen because of difficulties in finding a true state space representation [8]. This difficulty is caused by the bivariate of the transfer function and by its causality. The resulting matrix equation has two pseudo-state transition matrices.

Alexander [6] has shown that the recursive algorithm of (2.7) can be represented by the matrix recursive equation:

$$G_{m,n} = \bar{B}G_{m,n-1} + \bar{C}G_{m-1,n} + \bar{A}F_{m,n} \quad (3.1)$$

Where $G_{m,n}$ is a vector such that the elements of $G_{m,n}$ are the outputs $g(m-j,n-k)$ in (2.7) where $0 \leq j \leq L$ and $0 \leq k \leq L$. $F_{m,n}$ is a vector such that the elements of $F_{m,n}$ are the inputs $f(m-j,n-k)$ in (2.7) where $0 \leq j \leq L$ and $0 \leq k \leq L$. \bar{B} , \bar{C} and \bar{A} are appropriate coefficient matrices such that (2.7) and (3.1) are equivalent.

If the filter is unstable, then either \bar{B} , \bar{C} or $(\bar{B} + \bar{C})$ has at least one eigenvalue with a magnitude greater than or equal to one. Thus, stability analysis involves setting up the matrices \bar{B} and \bar{C} and finding the spectral radius of each matrix individually and of their sum.

4.0 SYNTHESIS

Often it is possible to express a desired two dimensional recursive digital filter as the product or sum of two one dimensional digital filters. That is, the ZW-Transform of the two dimensional filter may be expressed as the product

$$H(z,w) = H1(z)H2(w) \quad (4.1)$$

or as the sum

$$H(z,w) = H1(z) + H2(w) \quad (4.2)$$

In either case, the two dimensional synthesis problem is reduced to the synthesis of two one dimensional filters [9,10]. However, it is not possible to design sum separable or product separable digital recursive filters for all applications. For these applications where sum separable or product separable designs are not possible, the design of the required two dimensional digital recursive filter is considerably more complicated.

Many imaging systems have a natural circular symmetry. In general, the optical transfer function (OTF) of a circularly symmetric imaging system is circularly symmetric. Also, it is usually desirable to perform image processing where the processing is uniform with respect to direction. The natural

consequence is that filters with circularly symmetric impulse response functions are generally very desirable for image processing. A filter with a circularly symmetric impulse response is assured by restricting the Discrete Fourier Transform (DFT) for the filter to be circularly symmetric [11].

One popular method of designing digital recursive filters is to start with the Laplace Transform of the desired filtering function, make a suitable transformation to the Z-Transform domain and thus obtain the coefficients for the digital recursive filter. One such technique involves designing digital recursive filters from the squared magnitude characteristics of the desired filter which is really the squared magnitude of the Fourier Transform. This procedure is difficult to extend to two dimensions because of the difficulties encountered in factoring bivariate polynomials.

To illustrate this difficulty, consider the circularly symmetric Butterworth low pass filter squared magnitude characteristic.

$$H(r,s) = \frac{1}{1 + (-1)^n(r^2 + s^2)^n/R^2} \quad (4.3)$$

where r and s are the Laplace Transform variables for the x and y direction respectively and R is the desired radial cutoff frequency.

The bilinear transformation [9] can be used to obtain the corresponding recursive digital filter. First, we prewarp $H(r,s)$ to obtain

$$H(r,s) = \frac{1}{1 + a^{2n}(r^2 + s^2)^n} \quad (4.4)$$

where

$$a^2 = (-1)/\tan^2(RT/2) \quad (4.5)$$

(The assumption is made in this example that the sampling increment is the same in each direction and is equal to T .) Applying the bilinear transformation, we have an approximation for the ZW-Transform for the squared magnitude characteristic of the desired filter.

$$H(z,w) = \frac{1}{1 + a^{2n}[(z-1)/(z+1)]^2 + [(w-1)/(w+1)]^2} \quad (4.6)$$

If the polynomial in the denominator of (4.6) were factorable into distinct roots of z and w , then those roots would occur in reciprocal pairs. The design procedure would then be completed by forming $H(z,w)$ from those roots for which the magnitude of z is less than one and those for which the magnitude of w is less than one. The numerator polynomial of $H(z,w)$ is allowed to have roots with a magnitude of one.

$H(z,w)$ which is formed with the smaller in magnitude of each of the reciprocal pairs of roots in the numerator and denominator is said to have minimum phase. The minimum phase version of any filter is stable for any input sequence unless the denominator of its ZW-Transform has roots where either the magnitude of z or w is equal to one. In that case, it is conditionally stable.

However, the polynomial in the denominator of (4.6) is not factorable into distinct roots. Therefore, forming of the minimum phase version of $H(z,w)$ is not straightforward and the design procedure is not successful.

A minimum phase approximation to $H(z,w)$ can be obtained with the following procedure:

1. Construct the coefficient matrices \bar{B} and \bar{C} of (3.1) which corresponds to (4.6).
2. Calculate the eigenvalues of the matrix sum $(\bar{B} + \bar{C})$. They occur in reciprocal pairs.
3. Form the minimum phase approximation of the filter by using the smaller magnitude eigenvalue of each of the reciprocal pairs as a root of z and of w for the denominator polynomial and by using the minimum phase version of the numerator polynomial.

The resulting filter ZW-Transform is given by

$$H(z,w) = \frac{(1+p)^2(z+1)(w+1)}{4(z+p)(w+p)} \quad (4.7)$$

where

$$p = \frac{(2a - (2\sqrt{2})a + 1)}{1 - 2a} \quad (4.8)$$

5.0 FILTER DESIGN

5.1 Low Pass filter

Equation (4.7) gives the ZW-Transform for the low pass filter approximation which was derived from the circularly symmetric low pass filter squared magnitude characteristic of (4.3). For this particular design, the roots of $H(z,w)$ are real. In general, the roots may be real or they may occur in complex conjugate pairs. If the resulting filter is applied in a straightforward manner, the algorithm must handle complex numbers. This can be avoided by using a basic filter structure which uses only binomial functions resulting from the multiplication of two roots. When complex roots are involved, the pair of complex conjugate roots would form a basic filter stage. The penalty paid for this basic filter structure is that filters with odd numbers of zeros or poles can only be implemented by adding at least one null root. The addition of this null root results in unnecessary calculations in the algorithm which implements the filter. Thus, all filters designed will have the basic structure:

$$H(z,w) = \prod_{i=1}^4 \frac{A_i[z^2 + q(1i)z + q(2i)][w^2 + q(1i)w + q(2i)]}{[z^2 + p(1i)z + p(2i)][w^2 + p(1i)w + p(2i)]} \quad (5.1)$$

The basic low pass filter using this form is then given by

$$LP(z,w) = \frac{(1+p)^4(z^2 + 2z + 1)(w^2 + 2w + 1)}{16(z^2 + 2pz + p^2)(w^2 + 2pw + p^2)} \quad (5.2)$$

5.2 The Frequency Boost Filter

A frequency boost filter can be designed from the magnitude response characteristics of the low pass filter. Consider the filter which has a ZW-Transform given by:

$$H(z,w) = c + d|LP(z,w)|^2 \quad (5.3)$$

Note that (5.3) has roots of z and of w with magnitude greater than one since the roots occur in reciprocal pairs. This problem is overcome by using the minimum phase version of (5.3). Thus the ZW-Transform of the desired filter is given by:

$$H(z,w) = \frac{N(z,w)}{D(z,w)} \quad (5.4)$$

where $N(z,w)$ and $D(z,w)$ have minimum phase.

A high frequency boost filter can be designed by changing the values of c and d in (5.3). For the high pass filter, c has a value of one and d has a value of minus one. If a low frequency boost filter is desired with a magnitude gain of BF at DC and a gain of one at the Nyquist frequency, this can be achieved by setting:

$$\begin{aligned} c &= 1.0 \\ d &= -BF - 1.0 \end{aligned} \quad (5.5)$$

If a high frequency boost filter is desired with a magnitude gain of BF at the Nyquist frequency and a gain of one at DC, this can be achieved by setting:

$$\begin{aligned} c &= BF \\ d &= 1.0 - BF \end{aligned} \quad (5.6)$$

The shape of the resulting filter is also affected by the value of the root p which is derived from the design of the low pass filter. From (4.7) and (4.8), observe that p is a function of the desired radial cutoff frequency R , for the low pass filter. Note that three parameters, c , d and R , are required to design the filter specified by (5.3). However, if a high frequency boost or a low frequency boost filter is desired, then only two parameters, R and BF are required because c and d can be derived from BF .

6.0 FILTER DESIGN EXAMPLES

Figure 1 gives the perspective plot of a lowpass filter designed with the described technique with a cut off frequency which is 0.3 times the Nyquist frequency. Figure 2 gives the contour plot for this filter design. Figure 3 gives the perspective plot for a high frequency boost filter with a break frequency of 0.5 times the Nyquist frequency and a boost magnitude of 25.6. Figure 4 gives the contour plot for this filter design. Note that these examples are essentially circularly symmetric. Some degradation is observed as the break frequency approaches the Nyquist frequency. This is caused by the mapping characteristics of the bilinear transformation. Some degradation also occurs as the break frequency approaches DC. However, this can be corrected by using rotated filter combinations [12].

7.0 CONCLUSION

A design technique has been presented which can be used to design approximately circularly symmetric digital recursive filters for subjective image processing applications. These filters include lowpass, highpass, low and high frequency boost and bandpass filters. The filters are inherently stable because the denominator polynomial of the ZW-Transform is minimum phase.

REFERENCES

1. Ernest L. Hall, "A Comparison of Computations for Spatial Filtering", Proceedings of the IEEE, Vol. 60, no. 7, 1972, pp 887-891.
2. Lawrence R. Rabiner and Bernard Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1975, pp 442-455.
3. Samuel Stearns, Digital Signal Analysis, Hayden Publishing Co., Inc., Edison, N. J. p 134.
4. N. K. Bose, "Problems and Progress in Multidimensional System Theory", Proceedings of IEEE, Vol. 65, No. 6, 1977, pp 700-710.
5. Katsuniko Ogata, State Space Analysis of Control Systems, Prentice-Hall, Inc., Englewood Cliffs, N. J., p. 487.
6. Winzer E. Alexander and Steven A. Pruess, "Stability Analysis of Two Dimensional Digital Recursive Filters", accepted for publication in IEEE Transactions on Circuits and Systems.
7. Winzer E. Alexander, "Stability Analysis of Two Dimensional Digital Recursive Filters", 12th Asilomar Conference on Circuits, Systems, and Computers, November, 1978.
8. E. Fornasini and G. Marchesini, "State Space Realization Theory for Two Dimensional Filters", IEEE Transactions on Automatic Control, Vol. AC-21 1976, pp 484-492.
9. Bernard Gold and Charles Rader, Digital Processing of Signals, McGraw Hill, Inc., New York, N. Y., 1969.
10. Andreas Antoniou, Digital Filters: Analysis and Design, McGraw Hill, Inc., New York, N. Y., 1979.
11. Athanasios Papoulis, Systems and Transforms with Applications in Optics, McGraw Hill, Inc., New York, N. Y., 1968.
12. Dennis Goodman, "A Design Technique for Circularly Symmetric Low Pass Filters", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-26, No. 4, 1978, pp 290-304.

ORIGINAL PAGE IS
OF POOR QUALITY

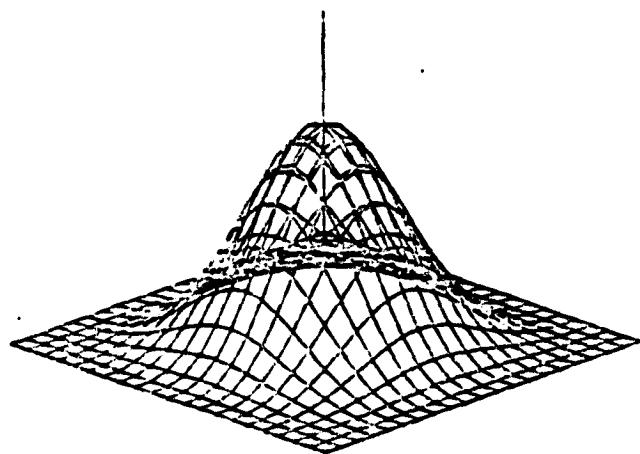


FIGURE 1. LOW PASS FILTER
STAGE = 1 $RC = 0.3$

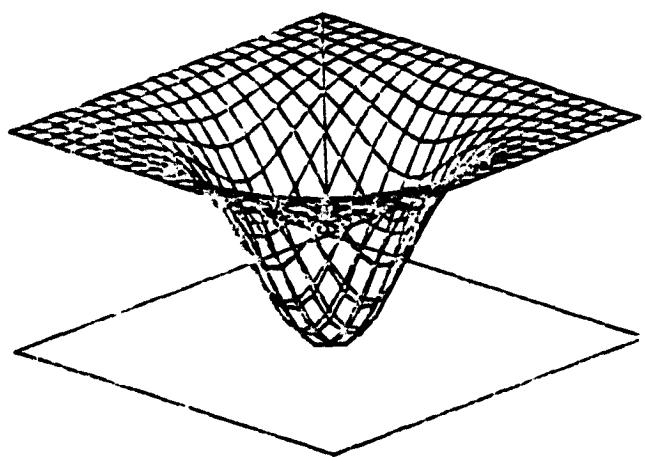


FIGURE 3. HIGH BOOST FILTER
BOOST FACTOR = 25.6 $RC = 0.5$

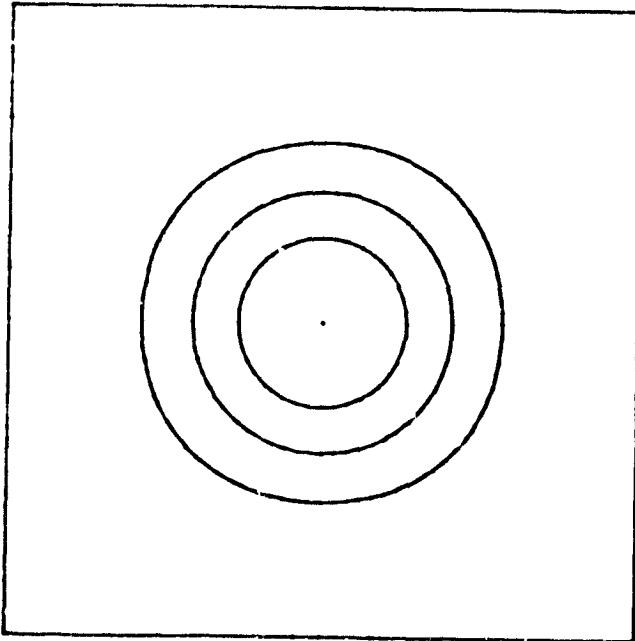


FIGURE 2. LOW PASS FILTER
STAGE = 1 $RC = 0.3$

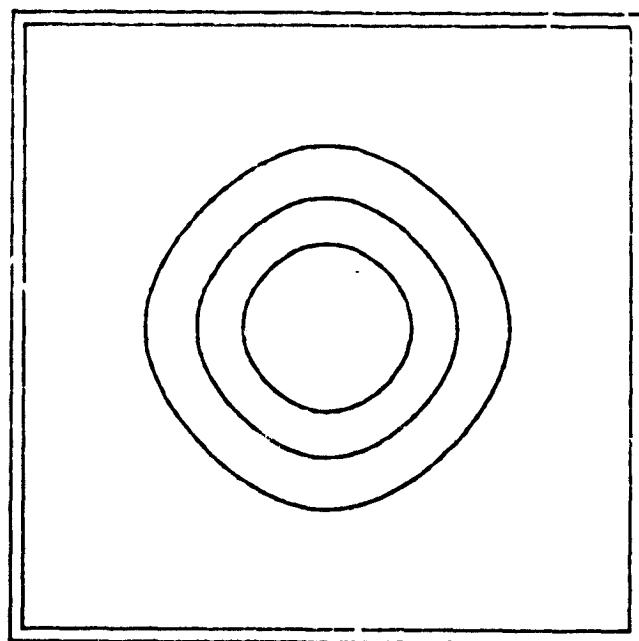


FIGURE 4. HIGH BOOST FILTER
BOOST FACTOR = 25.6 $RC = 0.5$

SIMULTANEOUS LINEAR ALGEBRAIC EQUATION FORMULATIONS OF 2D FILTERS

D. E. OLSON, W. E. ALEXANDER, E. E. SHERROD
Elec. Eng. Dept., NC A&T SU, Greensboro, NC

ABSTRACT

It is shown that 2D digital filter realizations are equivalent to the solution of tensor equations, and they are also equivalent to the solution of matrix equations. Both recursive and non-recursive filters are included in these formulations.

SUMMARY

A 2D digital filter, which possesses a rational transfer function, may be represented by its bivariate difference equation written in tensor form as:

$$b_{ij} g_{p+i,q+j} = a_{ij} f_{p+i,q+j} \quad (1)$$

where $1 \leq p \leq N$, $1 \leq q \leq M$, $-m \leq i \leq m$, $-m \leq j \leq m$; and the double appearance of an indice on a given side of the equality implying the usual summation over the appropriate range of that indice. A more formal expression of (1) is:

$$B_{pq}^{k1} g_{k1} = A_{pq}^{k1} f_{k1} \quad (2)$$

where $1 \leq k \leq N$, $1 \leq l \leq M$, and the non-zero components of the coefficient tensors given by $A_{pq}^{k1} = a_{k-p, l-q}$; and $B_{pq}^{k1} = b_{k-p, l-q}$; for $-m \leq k-p \leq m$, $-m \leq l-q \leq m$.

The 2D filtering operation requires that one determine all the g_{pq} , given all a_{ij} , b_{ij} , and f_{pq} . A solution will exist and be unique if there exists an inverse of the tensor B_{pq}^{k1} , say C_{uv}^{pq} ; with $1 \leq u \leq N$, $1 \leq v \leq M$. For such a case, the filtered solution would then be given by:

$$g_{uv} = C_{uv}^{pq} A_{pq}^{k1} f_{k1} \quad (3)$$

Tensor equation (2) can also be interpreted as a matrix equation with the A_{pq}^{k1} , B_{pq}^{k1} taken as $NM \times NM$ dimensional coefficient matrices with row index "pq", column index "k1"; and g_{k1} , f_{k1} taken as column matrices.

For the case when $N=M$, and $a_{00} \neq 0$, then equation (2) is also expressible as a matrix equation involving only $N \times N$ matrices given by:

$$LGR + \sum_{k=-m, k \neq 0}^m S_k G T_k = c PFQ + c \sum_{k=-m, k \neq 0}^m S_k F U_k \quad (4)$$

where $c = a_{00}/b_{00}$, the matrices $G = [g_{pq}]$, $F = [f_{pq}]$; and the non-zero components of the coefficient matrices L, R, P, Q, S_k , T_k and U_k given by:

(i) For p, q such that $-m \leq q-p \leq m$:

$$L_{pq} = b_{q-p, 0}/b_{00}; \quad R_{pq} = b_{0, q-p}/b_{00}; \quad P_{pq} = a_{q-p, 0}/a_{00}; \quad Q_{pq} = a_{0, q-p}/a_{00}.$$

$$T_{kpq} = b_{k, p-q}/b_{00} - b_{k, 0} b_{0, p-q}/b_{00}^2; \quad U_{kpq} = a_{k, p-q}/a_{00} - a_{k, 0} a_{0, p-q}/a_{00}^2.$$

(ii) And finally, for p, q such that $q-p=k$: $S_{kpq} = 1$.

Non-recursive filters generally require solutions of the form given by (3). For recursive filters (4) simplifies allowing solution by compact schemes.